

NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 14, Number 2

March, 1995

PRINT POST Approved - PP244099/00016



Sydney, New South Wales, Australia

\$3

TisHUG News Digest

All correspondence to:
C/o 3 Storey St.
Ryde 2112 Australia

TisHUG News Digest

ISSN 0819-1984

I N D E X

Title	Description	Author	Page No
The Board			
Co-ordinator			
Dick Warburton	(02) 918 8132		
Secretary			
Thomas Marshall	(02) 671 7535		
Treasurer			
Cyril Bohlson	(02) 639 5847		
Directors			
Percy Harrison	(02) 808 3181		
Loren West	(047) 21 3739		
Sub-committees			
News Digest Editor			
Loren West	(047) 21 3739		
BBS Sysop			
Ross Mudie	(02) 456 2122		
BBS telephone number	(02) 456 4806		
Merchandising			
Percy Harrison	(02) 808 3181		
Software Library			
Larry Saunders	(02) 644 7377		
Technical Co-ordinator			
Geoff Trott	(042) 29 8629		
BASCON	PROGRAM	STEPHEN SHAW	18
EXPANDING A TI99/4A FOR RS232/5 AND 6	HINTS	ROSS MUDIE	16
LEARN TO KNOW YOUR TI LESSON 24	TUTORIAL	PERCY HARRISON	10
REGIONAL GROUP REPORTS	GEN.INT.	VARIOUS	23/
TECHO TIME	TECHNICAL	GEOFF TROTT	12
THEME FROM STAR TREK	PROGRAM	G.RODDENBERRY AND A.COURAGE	6
TISHUG SHOP	CLUB NEWS	PERCY HARRISON	2
TISHUG SOFTWARE	PROGRAMS	LARRY SAUNDERS	3
TREASURER'S REPORT	REPORT	CYRIL BOHLSSEN	1
WONDER WORD PUZZLE	GAME	LOREN WEST	17

I B M I N D E X

PROGRAMMING IN BASIC	TIPS	B. V. TAKACH	19
----------------------	------	--------------	----

Regional Group Contacts

Central Coast	
Russell Welham	(043) 92 4000
Glebe	
Mike Slattery	(02) 692 8162
Hunter Valley	
Geoff Phillips	(049) 42 8176
Illawarra	
Geoff Trott	(042) 29 8629
Liverpool	
Larry Saunders	(02) 644 7377
Sutherland	
Peter Young	(02) 528 8775

Membership and Subscriptions

Annual Family Dues	\$35.00
Associate membership	\$10.00
Overseas Airmail Dues	A\$65.00
Overseas Surface Dues	A\$50.00

TisHUG Sydney Meeting

The March Meeting will start at
2.0 pm on the 4th March 1995
at Meadowbank Primary School,
Thistle Street, Meadowbank.

Printed by
Kwik Kopy West Ryde

TREASURER'S REPORT

by Cyril Bohlson

Income for previous month \$ 3487.00
Expenditure for previous month .. \$ 3242.08
Profit for previous month \$ 244.92
Membership accounted for \$ 35.00 of Income.
Shop sales \$ 3425.50 of Income.

The expenditure was made up of the following

Printing & Postage of TND \$ 274.16
BBS running cost \$ 69.80
Shop purchases \$ 2898.12



TISHUG SHOP.

with Percy Harrison.

The attendance at our February meeting (the first for 1995) was quite pleasing although a few familiar faces were missing such as Geoff Trott, who had a tennis commitment which prevented him from attending the meeting although he did pop in for ten minutes early in the morning to drop some repaired equipment back for the owners. Peter Schubert was very conspicuous by his absence, which was a little surprising as he had indicated to me a couple of weeks before the meeting that he had planned to be there so we hope that nothing serious has happened to Peter to keep him and his system away.

We were also a few systems down so the computer activities were lacking a bit although Larry and his complete system was once again to be seen as he managed to get the day off from work. Don Gould was also kept very busy on our club TI which, hopefully, will be upgraded to two double sided Half Height Drives by our next meeting.

The clubs PC was also set up and seemed to be kept busy with a game called Rapture although we did manage to wrestle it away for a while to demonstrate the IBM/TI emulator program to John Meldrum who apparently wants his PC screen to look like his old TI computer. This program has proven to be somewhat popular among our members who have taken the retrograde step of going over to the more readily available PC's.

This month I will once again include a list of the TI modules available from the shop in the hope that some of our newer members may want to add some of them to their collection.

TI MODULES

PRICE \$

Adventure Module and Cassette	12.00
Adventureland Cassette	4.00
Cart-writer Module	15.00
Car Wars Module	8.00
Disk Manager 2 Module	10.00
Early Learning Fun Module	8.00
Edu-pak Module	15.00
Homework Helper Module	15.00
Meteor Multiplication Module	8.00
Munch Man Module	8.00
Parsec Module	8.00
Personal Record Keeping Module	8.00
Reading On Module	8.00
Terminal Emulator II Module	15.00
TI Extended Basic Module	25.00
TI Invaders Module	8.00
TI Logo Module and Cassettes	15.00
Touch Typing Module	8.00
Tris Module	15.00
Tunnels of Doom Module and Cassette	10.00
Video Graphs Module	8.00

PC HARDWARE

PRICE \$

3 Button Mouse	14.00
Mouse Pad	3.00
101 Enhanced Keyboard	30.00
Nylon Screen Filter 14" Monitor	6.00
Glass Screen Filter 14" Monitor	20.00
Dustcover for 14" Monitor	7.00
Dustcover for Mini-tower	7.00
Dustcover for 80 Column Printer	7.00
Printer Stand	10.00
3.5 Floppy Drive Cleaning Kit	6.00
5.25 Floppy Drive Cleaning Kit	6.00
3.5 Diskette Storage Box (Holds 100)	10.00
5.25 Diskette Storage Box (Holds 100)	10.00
Printer Cable 6 Foot	6.00
Power Split Cable for 3.5 Drives	6.00
Power Split Cable for 5.25 Drives	6.00
3.5 Disk Drive 1.44Mb	58.00
CD Rom Drive 2 Speed	205.00
CD16 Discovery Pack	420.00
Sound/CD Speakers (Pair)	25.00
Super I/O Card (2S/1P/1G and IDE)	20.00

Above prices do not include postage.

A full range of IBM compatible PC hardware is available, including complete systems. Please contact me for all of your hardware requirements, our prices are very competitive as we do not have any overheads.

Bye for now.

TI sHUG SOFTWARE **BY LARRY SAUNDERS**

PAGE PRO IMPORT PREVIEW

Previews files created with the WYSIWYG Columnizer.

Diskname P105
 Used= 344 Free= 16

This is GOFER, by Dan Gazsy, it is a must for any one who uses PAGE PRO.

PROGRAM OVERVIEW

FCTN - 1 = Deletes character.
 FCTN - 2 = Turns ON Insert Mode.
 FCTN - 3 = Erases the string from the cursur.
 FCTN - 6 = Returns to the starting position.

MEMORY IMAGE FILE LOADER

Will load Page Pro 99, or any other Image file from disk drives 1 to 9.

CATALOG

Will catalog any disk/ram/hard drive, 1 to 9.

PAGE PRO TO INSTANCE CONVERTER

Converts Page Pro pictures to TI-Artist Instances, catalogs the disk first, then you can select (S) or deselect (D), when selecting is compeleted, press Q.

INSTANCE TO PAGE PRO CONVERTER

Converts TI-Artist Instances to Page Pro Pictures.

FIX128 <<-->> VAR80 TEXT CONVERSION

Converts either direction, primarily to aid in converting IBM text files over to the TI format.

MODIFY PAGE PRO PAGE FILES

Allows you to reassign drives for part or all pictures in Page Pro page file.

WYSIWYG COLUMNIZER

For columnizing, creates two-column output files of 66 lines for Page Pro.

ANALYZE PAGE PRO FILE

This option allows you to inentify the picture files associated with a particular PAGE file.

ANALPAGE	23 Prog	CATALOG	5 Prog
CONFIG	4 d 80	CURSOR	5 Prog
CUSTOMIZE	8 Prog	FIX2VAR	18 Prog
GOFER	33 Prog	GOFERXB	17 D 80
GOFES	33 Prog	I2PCONV	33 Prog
LOAD	2 Prog	MENU1	4 d 80
MENU2	3 d 80	MENU6	4 d 80
MIFLDR	8 Prog	MODPAGE	24 Prog
LOWMEM	33 Prog	P2ICONV	33 Prog
PPSCR1	54 Prog		

Disk#1:P106
 Used= 354 Free= 4

GOFER DOCUMENTATION ADDENDUM

ADDITIONAL MODULES:

Since version 1.0 of GOFER was released, I've come up with a batch PCX to PP converter which no other TI program currently performs. Like all the other GOFER modules, it is dependent on the GOFER kernel being resident when invoked. It can be invoked in the same manner as any function of GOFER (see previous docs for loading).

Once selected, you are prompted to specify where to search (disk drive) for PCX files. Once a drive is selected, a catalog of all picture files is created and placed in a scrolling window. Program can only handle a max of 25 PCX files at a time. If their are more than 25 on the source drive, they will be ignored. If no drive is selected, the window disappears and you are back at the main menu.

Once in the scrolling window mode, use the E/X keys to move through the catalog. The S/D keys are used to select/deselect a file for conversion. Selecting a file will automatically copy the input file as the suggested output file. You have the option to override the suggested name, refer to Keyboard Input Addendum for additonal cursor control. The Q key terminates the selection process. Last you are asked to identify the output drive for the pictures. Once this is satisfied, The window clears and it becomes a status window. The number of files to convert, the current input/output files and the state of the conversion will be displayed during the conversion process. Upon completion you'll be returned to the initial prompt which asked you to identify the source drive.

Once the conversion process commences, it will analyze the PCX header and display some of the vital stats it contains. The most important items are: the horizontal and vertical size (in pixels) and the type (number of planes in the picture). In addition, you'll

also be told what version of PC Paintbrush created the file. While Page Pro will only let you view 60 columns (480 pixels) across the page, many a PCX file will exceed that bound. This module will permit you to convert PCX files as wide as 150 columns (1800 pixels). There is no limitation on the vertical size of the file because of the manner used to convert the PCX. This program is not capable of dithering or half-toning (a process of converting color images to black and white), so any PCX file with more than one plane will not convert. Besides, this utility was intended to only port clipart over to Page Pro picture format. If you attempt to supply it with the identical name (drive and filename) for both the source and destination file, you'll receive an error.

After the first PCX data record is read, you'll see the output dimensions of the Page Pro picture displayed in the output file area. As reads and writes are performed, brief messages will appear in the status area of the screen. If an error should occur, you'll get a message in the error section of the screen. Once you acknowledge the error (by pressing the ENTER key), the program will close all files and resume with the next conversion.

Note: PCX files will generally yield quite large Page Pro picture files. The average number of converted PCX's that will fit on a D/S D/D disk are 10-15 files.

If there is a question as to it's marketability, just look at the type of files that Ben Yates DEZIP unpacks. You'll quickly see that much of it is either in TIFF, IMG, GIF or PCX format. BTW, the memory image loader (MIFLDR) will load the un-registered copy of DEZIP with no problems. Can't say the same for FunWeb 4.31.

The registered copy of DEZIP isn't as forgiving. In an effort to extend the DEZIP program to XB cartridge owners, Ben Yates has included the E/A utils resident in low memory as a loadable disk file for his program. However, he chose to append this file to the very end (which seems reasonable). After inspecting DEZIP, it was discovered that one of DEZIP's modules resides very high in extended memory (area used both by GOFER and FW). If this troublesome module were the last one to load, neither program would have a problem loading. Since this is not the case, the loader is overwritten before the program load process completes.

PCX2PP	33 Prog	PCX-PIC01	29 D128
PCX-PIC02	38 D128	PCX-PIC03	57 D128
PCX-PIC04	30 D128	PCX-PIC05	37 D128
PCX-PIC06	33 D128	PCX-PIC07	32 D128
PCX2PPDOCS	21 d 80	PREVIEW	17 Prog
WYSIWYG	27 Prog		

Diskname U107
Used= 358 Free= 0

Line Editor program, this program will edit any DV80 file no matter what size it is.

CHARA1	9 Prog	CHARA3	296 D 80
EXEC	21 Prog	LINEDITOR	32 Prog

Disk#1:U108
Used= 247 Free= 111

SOUNDMAKER

Programmed by Jim Peterson
Dec. 1984 - copyright 1984 Tigercub Software,

This program will help you to develop sound effects. The default values for each option will be those chosen, the previous time, so that you can easily experiment with modifying a sound. For instance, you can use the random option until you hear an interesting effect, then switch to selective to refine it. When you have perfected the effect, use the selective and save options to save it to disk as a MERGE format file. The MERGE file can then be converted to a RUNable program, or MERGE'd into another program, by MERGE DSK1.(filename) If you select 4 tones, remember that one of them must be a noise tone between -1 and -8. For bass tones, select the 4 note option. Give the 1st and 2nd notes a positive value; the frequency and volume may be either audible or inaudible. Give the 3rd note a positive value below 1000 and a volume of 30, and the 4th note a 4 noise with an audible volume.

TYPE WRITER

This is a very simple very easy to use typewriter program.

MAGICFM

A program along the lines of Mass Transfer.


CARD

TI Cardfile is a index card program, a database with only two fields: the title, or index, and the text. Each file has space for about 275 files each.

SOUND EXPERIMENT

A sound making program.

CARD	6 Prog	CARDEDIT	22 Prog
CARDFILE	27 Prog	CARDUTIL	24 Prog
CHARA1	9 Prog	LOAD	5 Prog
LOAD/MAGIC	34 Prog	MAGICFM	20 Prog
ROOT	28 Prog	SOUND/EXP	15 Prog
SOUNDMAKER	37 Prog	TYPE	20 Prog

 END OF ARTICLE

TiSHUG SOFTWARE BY LARRY SAUNDERS

Diskname AT111
Used= 333 Free= 25

Diskname P109
Used= 353 Free= 5

Page Pro FX (EFFECTS)

This utilities disk will enlarge, reduce, invert, or "ghost" Page Pro pics.

Flipper

This utility is used to mirror pics side by side or up and down.

Rotation

This utility allows a Page Pro Picture to be rotated 90 degrees at a time.

Stripper

This utility removes unused white areas around a picture, thus making the picture smaller.

Font Editor

This utility you can design or customize your own Line Fonts, Small Fonts, or Large Fonts. You can load in any Page Pro Font (except Page Pro Headline Fonts).

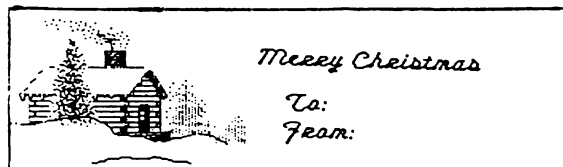
The Font Editor will allow you to alter a old Font or make a new font

CARPENTER	33 I 13	HAPPYSUN	9 I 13
LOAD	15 Prog	MADHATTER	34 I 13
PPEDIT	33 Prog	PPEDIU	31 Prog
PPEFX-V100	66 i254	PPF-V110	26 Prog
PPR-V100	51 i254	PPS-V100	16 Prog
TVREPAIRMN	39 I 13		

Diskname P110
Used= 351 Free= 7

Page Pro Pictures

The first two are pictures, XMASLB is a picture full of Christmas Labels, Example listed below.



ILOVEYOU	33 I 13	VALEN	140 I 13
XMASLB	178 I 13		

Artist Font Maker

This utility allows you to make Artist Fonts, Modified a Font, or make a new font. It will allow you to make a new Font file or Append to a old font file. This program has three prompts only, the first one is the file that you want to make the Font from (e.g. APPLE18), the second is the name of the Font you want it saved to, the third prompt is what character it is. Make sure that you press the right character, if it is for e.g. a CAPITAL 'M' press SHIFT M if ALPHA LOCK is not lock down.

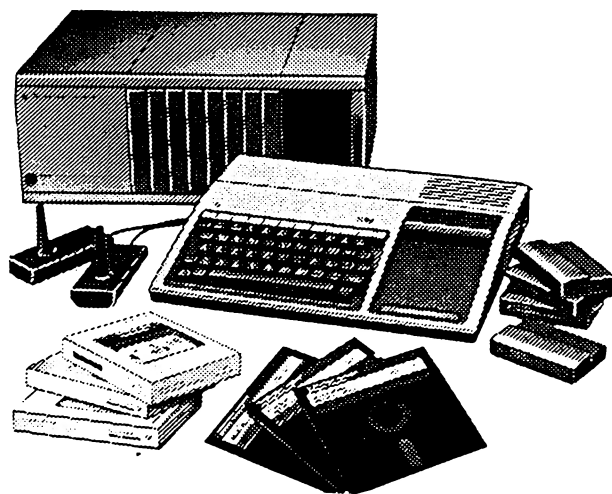
AFM	16 Prog	ALDERNEY20	25 Prog
APPLE18	25 Prog	ATHENS18	25 Prog
CHANCERY18	25 Prog	CHARA1	9 Prog
CLAIRVX24	25 Prog	DURANGO40	25 Prog
EON12	25 Prog	HU24	25 Prog
LOAD	5 Prog	MIKE24A	25 Prog
MIKE24B	25 Prog	ROOT	28 Prog
SHADOW48	25 Prog		

Diskname P112
Used= 302 Free= 56

Page Pro Pictures

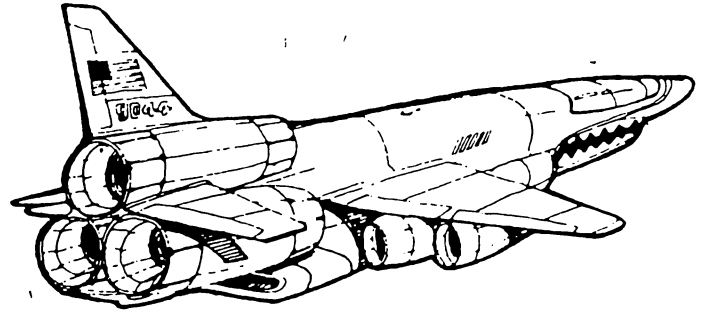
HF	102 I 13	HEART1	33 I 13
HLANE1	100 I 13	HLANE2	67 I 13

END OF ARTICLE



THEME FROM STAR TREK

This program plays the theme from Star Trek whilst displaying graphic pictures of the two leaders, Captain Kirk and Mr. Spock, it also displays the words to the music as it plays, it is quite enjoyable to watch and listen.



If anybody would like a copy of this program, if you have'nt the time to type it in please get in touch with me at one of the meetings. (ED)

```

5 CALL CHAR(96,"0000201008")
10 GOTO 100
11 A,B,C,D,E,F,G,X
30 CALL PICTURE :: CALL SOUN
D :: CALL BEAMMEUP :: CALL C
OLOR :: CALL BEAMDOWN :: CAL
L KEY :: CALL CLEAR
40 !@P-
100 ! *****
110 ! * Theme from *
120 ! * "S T A R T R E K" *
130 ! * TV Series Version *
140 ! * by GENE RODDENBERRY*
150 ! * and ALEX COURAGE *
160 ! *****
170 ! Programmed in Extended
Basic by Ken Gilliland
180 ! Send comments to: 543
Riverdale #15, Glendale, CA
91204
190 CALL PICTURE :: GOTO 230
200 CALL SOUND(A,B,C,D,E,F,G
):: RETURN
210 A=A/2
220 CALL SOUND(-A,B,C,D,E,F,
G):: CALL SOUND(-A,B*1.003,C
,D*1.007,E,F,G):: RETURN
230 FOR X=1 TO 2 :: A=600 ::
B=392 :: C=30 :: D=220 :: E=
30 :: F=131 :: G=25 :: GOSUB
200 :: C=20 :: E=20 :: GOSUB
200 :: F=196 :: GOSUB 200 ::
GOSUB 200
240 C=30 :: E=30 :: F=262 ::
GOSUB 200 :: C=20 :: E=20 ::
GOSUB 200 :: A=300 :: F=131
:: GOSUB 200 :: C=30 :: GOSU
B 200 :: A=600 :: C=20 250 D
=294 :: GOSUB 200 :: C=30 ::
E=30 :: GOSUB 200 :: B=392 :
: C=20 :: D=208 :: E=20 :: G
OSUB 200 :: F=196 :: GOSUB 2
00 :: GOSUB 200 :: C=30 :: E
=30
260 F=262 :: GOSUB 200 :: C=
20 :: E=20 :: GOSUB 200 :: A
=300 :: F=131 :: GOSUB 00 ::
C=30 :: GOSUB 200 :: A=600 :
: C=20 :: D=277 :: GOSUB 200
:: NEXT X
270 CALL BEAMMEUP :: FOR A=9
TO 12 :: CALL COLOR(A,14,2):
: NEXT A :: GOSUB 280 :: GOSU
B 280 :: GOTO 1130
280 DISPLAY AT(20,5):"be" ::
A=1000 :: B=392 :: C=30 :: D
=196 :: E=15 :: F=131 :: G=20
:: GOSUB 210 :: DISPLAY AT(2
0,7):"yond"
290 A=500 :: C=10 :: GOSUB 2
10 :: F=196 :: GOSUB 220 ::
GOSUB 220 :: E=30
300 B=349 :: F=262 :: GOSUB
220 :: E=15 :: GOSUB 220 ::
A=250 :: F=196 :: GOSUB 210
:: GOSUB 220 :: A=500 :: D=2
62 :: GOSUB 210
310 A=1000 :: E=30 :: F=131
:: GOSUB 210 :: DISPLAY AT(2
0,12):"the" :: B=330 :: F=26
2 :: GOSUB 220
320 DISPLAY AT(20,16):"rim"
:: A=666 :: B=294 :: F=196 :
: GOSUB 210 :: A=334 :: B=26
2 :: GOSUB 210 :: DISPLAY AT
(20,20):"of"
330 F=131 :: GOSUB 220 :: DI
SPLAY AT(20,23):"the" :: A=6
66 :: B=247 :: GOSUB 210 ::
DISPLAY AT(21,10):"star"
340 A=500 :: B=185 :: C=30 :
: D=233 :: E=10 :: F=208 ::
G=27 :: GOSUB 210 :: C=1 5
350 GOSUB 220 :: F=156 :: G=
20 :: GOSUB 220 :: DISPLAY A
T(21,14):"light" :: GOSU B 2
20 :: C=30
360 F=208 :: GOSUB 220 :: C=
15 :: GOSUB 220 :: A=250 ::
F=156 :: GOSUB 210 :: GO SUB
220 :: A=500 :: B=262 :: GOS
UB 210
370 A=500 :: B=185 :: C=30 :
: D=233 :: E=10 :: F=208 ::
G=27 :: GOSUB 210 :: C=1 5 :
: GOSUB 220 :: F=156 :: G=20
:: GOSUB 220 :: GOSUB 220 ::
C=30
380 F=208 :: GOSUB 220 :: C=
15 :: GOSUB 220 :: A=250 ::
F=208 :: G=27 :: GOSUB 10 ::
GOSUB 220 :: A=500 :: GOSUB
210 :: DISPLAY AT(20,3):"ny"
:""
390 A=500 :: B=392 :: C=30 :
: D=196 :: E=15 :: F=131 ::
G=20 :: GOSUB 210 :: DIS PLA
Y AT(20,6):"love" :: C=10 ::
GOSUB 220 :: F=196 :: GOSUB
220 :: GOSUB 220 :: E=30
400 B=392 :: D=220 :: F=262
:: GOSUB 220 :: E=15 :: GOSU
B 220 :: A=250 :: F=196 :: G
OSUB 210 :: GOSUB 220 :: A=5
00 :: D=294 :: GOSUB 210

```

```

410 A=1000 :: E=30 :: F=131
:: GOSUB 210 :: DISPLAY AT(2
0,11):"is" :: B=349 :: =262
:: GOSUB 220 :: DISPLAY AT(2
0,14):"wan"
420 A=666 :: B=330 :: F=196
:: GOSUB 210 :: DISPLAY AT(2
0,17):"d'ring" :: A=334 :: B
=294 :: GOSUB 210
430 F=131 :: GOSUB 220 :: DI
SPLAY AT(20,24):"in" :: A=66
6 :: B=262 :: GOSUB 210 :: D
ISPLAY AT(21,9):"star"
440 A=500 :: B=196 :: C=30 :
: D=247 :: E=15 :: F=156 ::
G=20 :: GOSUB 210 :: C=1 0 :
: GOSUB 220 :: F=220 :: GOSU
B 220 :: GOSUB 220 :: C=30 :
: DISPLAY AT(21,13):" flight
450 F=311 :: GOSUB 220 :: E=
10 :: GOSUB 220 :: A=250 ::
F=220 :: GOSUB 210 :: GO SUB
220 :: A=500 :: B=277 :: GOS
UB 210
460 C=30 :: F=156 :: GOSUB 2
20 :: B=196 :: C=15 :: GOSUB
220 :: F=311 :: GOSUB 20 ::
GOSUB 220 :: A=1000 :: F=220
:: GOSUB 210 :: C=30
470 DISPLAY AT(20,2):" i":"
480 F=156 :: GOSUB 220 :: DI
SPLAY AT(20,7):"know" :: A=5
00 :: B=370 :: D=220 :: F=14
7 :: G=27 :: GOSUB 210 :: C=
15 :: GOSUB 220 :: F=110 ::
G=20 :: GOSUB 220 :: GOSUB 2
20
490 C=30 :: F=147 :: GOSUB 2
20 :: C=15 :: GOSUB 220 :: D
ISPLAY AT(20,12):"he'll" ::
A=1000 :: C=30 :: D=247 :: F
=110 :: GOSUB 210
500 DISPLAY AT(20,18):"find"
510 D=277 :: F=147 :: GOSUB
220 :: DISPLAY AT(20,23):"in
" :: D=294 :: F=294 :: OSUB
220 :: DISPLAY AT(21,3):"sta
r"
520 A=666 :: D=330 :: F=220
:: GOSUB 210 :: DISPLAY AT(2
1,8):"clus" :: A=334 :: D=37
0 :: GOSUB 210 :: F=147 :: G
OSUB 220
530 DISPLAY AT(21,12):"tered
" :: A=666 :: D=392 :: GOSUB
210 :: DISPLAY AT(21,19 ):"r
each" :: A=500 :: D=440 :: F
=139 :: GOSUB 210 :: B=247 :
: C=15 :: GOSUB 220

```

```

540 F=196 :: GOSUB 220 :: GO
SUB 220 :: DISPLAY AT(21,24)
:"es"
550 C=30 :: D=466 :: F=277 :
: GOSUB 220 :: C=15 :: GOSUB
220 :: A=250 :: F=196 :: GOSU
B 210 :: GOSUB 220 :: A=500
:: B=311
560 GOSUB 210 :: C=30 :: F=1
39 :: GOSUB 220 :: B=247 ::
C=15 :: GOSUB 220 :: F=2 77
:: GOSUB 220 :: GOSUB 220 ::
C=30 :: F=196 :: GOSUB 220 :
: C=15 :: GOSUB 220
570 A=250 :: F=139 :: GOSUB
210 :: GOSUB 220 :: A=500 ::
B=311 :: GOSUB 210
580 DISPLAY AT(20,3):" love
":" :: C=30 :: D=233 :: F=1
56 :: GOSUB 220 :: B=262 ::
C=15 :: GOSUB 220 :: F=233 :
: GOSUB 220
590 GOSUB 220 :: DISPLAY AT(
20,10):"strange" :: C=30 ::
F=311 :: GOSUB 220 :: C= 15
:: GOSUB 220 :: A=1000 :: C=
30 :: D=262 :: F=233 :: GOSU
B 210
600 DISPLAY AT(20,18):"love"
610 D=294 :: F=156 :: GOSUB
220 :: DISPLAY AT(20,23):"a"
:: D=311 :: F=311 :: GO SUB
220 :: DISPLAY AT(21,6):"sta
r"
620 A=666 :: D=349 :: F=277
:: GOSUB 210 :: DISPLAY AT(2
1,10):"wo" :: A=334 :: =392
:: GOSUB 210 :: F=156 :: GOS
UB 220 :: DISPLAY AT(21,12):
"nan"
630 A=666 :: D=415 :: GOSUB
210 :: DISPLAY AT(21,16):"te
ach" :: A=500 :: D=466 :: F=1
96 :: GOSUB 210 :: B=247 ::
C=15 :: GOSUB 220 :: F=392 :
: GOSUB 220
640 GOSUB 220 :: DISPLAY AT(
21,21):"es" :: C=30 :: D=494
:: F=196 :: GOSUB 220 :: C=15
:: GOSUB 220 :: A=250 :: G=2
7 :: GOSUB 210 :: GOSUB 220
:: A=500 :: B=311
650 GOSUB 210 :: C=30 :: G=2
0 :: GOSUB 220 :: B=247 :: C
=15 :: GOSUB 220 :: F=39 2 :
: GOSUB 220 :: GOSUB 220 ::
C=30 :: F=196 :: GOSUB 220
660 C=15 :: GOSUB 220 :: A=2
50 :: G=27 :: GOSUB 210 :: G

```

```

OSUB 220 :: A=500 :: B=3 11
:: GOSUB 210
670 DISPLAY AT(20,3):" i":"
:: C=30 :: D=196 :: F=131 ::
GOSUB 220
680 B=294 :: C=15 :: GOSUB 2
20 :: F=196 :: GOSUB 220 ::
GOSUB 220 :: B=196
690 DISPLAY AT(20,8):"know"
:: C=30 :: D=349 :: F=262 ::
GOSUB 220 :: C=15 :: GO SUB
220
700 A=250 :: F=196 :: GOSUB
210 :: GOSUB 220 :: A=500 ::
B=262 :: GOSUB 210 :: =1000
:: C=30 :: F=131 :: GOSUB 21
0 :: D=311 :: F=262
710 DISPLAY AT(20,14):"his"
720 GOSUB 220 :: DISPLAY AT(
20,18):"jour" :: A=666 :: D=
294 :: F=196 :: GOSUB 22 0
730 DISPLAY AT(20,22):"ney"
:: A=334 :: D=262 :: GOSUB 2
10 :: F=131 :: GOSUB 220 ::
DISPLAY AT(21,10):"ends" ::
A=666 :: D=247 :: GOSUB 210
740 DISPLAY AT(21,15):"nev"
750 A=500 :: B=185 :: C=30 :
: D=233 :: F=208 :: G=27 ::
GOSUB 210 :: C=15 :: GOS UB
220 :: F=156 :: G=20 :: GOSU
B 220 :: GOSUB 220 :: C=30
760 DISPLAY AT(21,18):"er"
770 F=208 :: GOSUB 220 :: C=
15 :: GOSUB 220 :: A=250 ::
F=156 :: GOSUB 210 :: GO SUB
220 :: A=500 :: B=262 :: GOS
UB 210
780 B=185 :: C=30 :: F=208 :
: G=27 :: GOSUB 220 :: C=15
:: GOSUB 220 :: G=20 :: GOSU
B 220 :: GOSUB 220 :: C=30 :
: F=156 :: GOSUB 220 :: C=15
790 GOSUB 220 :: DISPLAY AT(
20,3):" his":"
800 A=1000 :: C=30 :: D=208
:: F=208 :: G=27 :: GOSUB 21
0 :: DISPLAY AT(20,9):"s tar
" :: A=500 :: B=330 :: D=196
:: F=131 :: GOSUB 210 :: C=1
5 :: GOSUB 220
810 F=196 :: GOSUB 220 :: GO
SUB 220 :: DISPLAY AT(20,14)
:"trek" :: B=220 :: C=30 ::
D=392 :: F=262 :: GOSUB 220
:: C=15 :: GOSUB 220 :: A=25
0
820 F=196 :: GOSUB 220 :: GO
SUB 220 :: A=500 :: B=294 ::

```



```

GOSUB 210 :: A=1000 :: C=30
:: F=131 :: GOSUB 210 :: DIS
PLAY AT(20,19):"will" :: D=3
49 :: F=262 :: GOSUB 210
830 DISPLAY AT(20,24):"go"
840 A=666 :: D=330 :: F=196
:: GOSUB 210
850 DISPLAY AT(21,10):"on" :
: A=334 :: D=294 :: GOSUB 21
0 :: F=131 :: GOSUB 220 :: D
ISPLAY AT(21,13):"for" :: A=
666 :: D=262 :: GOSUB 210
860 DISPLAY AT(21,16):"ev"
870 A=500 :: B=220 :: D=247
:: F=175 :: GOSUB 210 :: C=1
5 :: GOSUB 220 :: F=349 :: G
OSUB 220 :: GOSUB 220 :: C=3
0 :: F=175 :: GOSUB 220 :: C
=15
880 GOSUB 220 :: A=250 :: GO
SUB 210 :: GOSUB 220 :: A=50
0 :: B=311 :: GOSUB 210 :: D
ISPLAY AT(21,18):"er" :: B=2
08 :: C=30 :: F=165 :: GOSUB
220 :: C=15 :: GOSUB 210 ::
F=330 :: GOSUB 220
890 GOSUB 220 :: C=30 :: F=1
65 :: GOSUB 220 :: C=15 :: G
OSUB 220 :: DISPLAY AT(20,3
):"but"." " :: A=1000 :: C=30
:: GOSUB 210
900 DISPLAY AT(20,7):"tell"
910 A=500 :: B=262 :: D=220
:: F=175 :: GOSUB 210 :: C=1
5 :: GOSUB 220 :: F=262 :: G
OSUB 220 :: GOSUB 220 :: C=3
0 :: F=349 :: GOSUB 220
920 C=15 :: GOSUB 220 :: DISPLAY
AT(20,12):"him" :: A=1000 ::
C=30 :: D=247 :: F=175 :: GOS
UB 210 :: DISPLAY AT(20,16):
"while" :: B=208 :: C=15 ::
D=262 :: F=233 :: GOSUB 220
:: D=294 :: F=175
930 DISPLAY AT(20,22):"he"
940 GOSUB 220 :: DISPLAY AT(
21,3):"wan" :: A=666 :: D=33
0 :: F=233 :: GOSUB 210
950 DISPLAY AT(21,6):"ders"
:: A=334 :: D=349 :: GOSUB 2
10 :: F=117 :: GOSUB 220 ::
DISPLAY AT(21,11):"his" :: A
=666 :: B=330 :: GOSUB 210
960 DISPLAY AT(21,15):"star"
970 A=500 :: B=220 :: C=30 :
: D=392 :: F=131 :: GOSUB 21
0 :: C=15 :: GOSUB 220 :: F=2
62 :: GOSUB 220 :: GOSUB 220

```

```

:: C=30 :: F=196 :: GOSUB 22
0
980 C=15 :: GOSUB 220 :: DIS
PLAY AT(21,19):"ry" :: A=250
:: F=131 :: GOSUB 210 :: GOSU
B 220 :: A=500 :: B=294 :: G
OSUB 210
990 DISPLAY AT(21,22):"sea"
1000 B=233 :: C=30 :: D=466
:: F=110 :: GOSUB 220 :: C=1
5 :: GOSUB 220 :: F=220 :: G
OSUB 220 :: GOSUB 220 :: C=3
0 :: GOSUB 220 :: C=15
1010 GOSUB 220 :: DISPLAY AT(20,3
):" re"." "
1020 A=250 :: D=440 :: GOSUB
210 :: GOSUB 220 :: A=500 ::
B=277 :: GOSUB 220 :: DISPLA
Y AT(20,7):"men"
1030 B=220 :: C=30 :: D=392
:: F=147 :: GOSUB 220 :: C=1
5 :: GOSUB 220 :: F=220 :: G
OSUB 220 :: GOSUB 220 :: DIS
PLAY AT(20,10):"ber" :: C=30
:: D=262 :: F=294 :: GOSUB 22
0
1040 C=15 :: GOSUB 220 :: A=
250 :: F=147 :: GOSUB 210 ::
GOSUB 220 :: A=500 :: F=349 ::
: GOSUB 210
1050 C=30 :: F=196 :: GOSUB
220 :: C=15 :: GOSUB 220 ::
A=1000 :: F=147 :: GOSUB 210
:: DISPLAY AT(20,14):"re" ::
A=666 :: B=247 :: D=294 :: F
=196 :: GOSUB 210 :: A=334
1060 DISPLAY AT(20,16):"mem"
1070 GOSUB 210 :: F=196 :: G
=27 :: GOSUB 220 :: DISPLAY
AT(20,19):"ber" :: A=666 ::
GOSUB 210 :: DISPLAY AT(20,2
3):"me" :: A=500 :: B=220 ::
C=30 :: D=262 :: F=131 :: G
OSUB 210
1080 C=15 :: GOSUB 220 :: F=
196 :: GOSUB 220 :: GOSUB 22
0 :: C=30 :: F=262 :: GOSUB
220 :: C=15 :: GOSUB 220 ::
A=250
1090 F=131 :: GOSUB 210 :: G
OSUB 220 :: A=500 :: B=294 :
: GOSUB 210
1100 B=208 :: C=30 :: E=13 :
: F=131 :: GOSUB 210 :: DISP
LAY AT(20,1):" "
1110 C=15 :: E=17 :: GOSUB 2
20 :: F=196 :: E=23 :: GOSUB
220 :: E=27 :: GOSUB 220 ::

```

```

C=30 :: F=262 :: E=29 :: GOS
UB 220 :: C=15 :: GOSUB 220
:: A=250
1120 F=131 :: GOSUB 210 :: G
OSUB 220 :: A=500 :: B=294 :
: GOSUB 210 :: RETURN
1130 A=1000 :: B=220 :: C=15
:: D=392 :: E=15 :: F=262 ::
GOSUB 210 :: G=30 :: GOSUB 22
0 :: F=196 :: G=25 :: GOSUB
220 :: G=30 :: GOSUB 220
1140 F=262 :: G=25 :: GOSUB
220 :: FOR A=1 TO 200 :: NEX
T A :: CALL BEAMDOWN
1150 FOR A=9 TO 12 :: CALL C
OLOR(A,14,1): NEXT A :: DIS
PLAY AT(1,3):"theme from the
tv series": " star trek"
1160 DISPLAY AT(5,2):"music
by alexander courage": " ly
rics by gene roddenberry " :
: CALL BEAMMEUP
1170 DISPLAY AT(20,2):"a pro
gram by ken gilliland": : :
:" want to hear it again"
1180 CALL KEY(0,A,B):: IF B=
0 THEN 1180
1190 CALL BEAMDOWN :: IF A=8
9 THEN 100 ELSE IF A=121 THE
N 100 ELSE CALL CLEAR :: RUN
"DSK1.LOAD"
1199 !@P+
1200 SUB PICTURE
1204 GOTO 1210
1205 A,A$,B
1207 CALL CLEAR :: CALL SCRE
EN :: CALL COLOR :: CALL CHA
R :: CALL HCHAR :: CALL VCHA
R
1208 !@P-
1210 CALL CLEAR :: CALL SCRE
EN(2):: FOR A=0 TO 14 :: CAL
L COLOR(A,2,2):: NEXT A
1220 RESTORE 1240 :: FOR A=1
37 TO 143 :: READ A$ :: CALL
CHAR(A,A$):: NEXT A
1230 FOR A=40 TO 88 :: READ
A$ :: CALL CHAR(A,A$):: NEXT A
1239 !@P+
1240 DATA 0000000000000101,0
000071F7F7FFFF,0000E0FCFFFF
FFFF,00000000000080C0C,000 000
000001050F
1241 !@P-
1250 DATA 0000003E7F6B963B,0
0000000B0F 8FCFE,00,03030707
07070504,FF FFFFFFFE0E0DEC7
1260 DATA FFFFFFFE0000078E,C

```

```

EOEOEOEO60607,041F091B131F3
A1B,F51FFFF0F0C08,FF7F3F 3F1
FOFOFOF
1270 DATA 0000000000000808,04
00020202000002,CA80814000389
1C2,D598141000936804,404 040
4040A0C08
1280 DATA 181B0D1A08040404,0
7D8C7AA0000404E,8D4E8C4E0A14
030F,80808080808,0100000 000
000003
1290 DATA 4242615051C9E4F3,0
4F40802A21 408F2,80,02020201
0100010A,31 809CBF408C9260
1300 DATA 224202029313272F,0
000000080C0A0D8,0C708F002F00
E7FF,381C0F87F960F8FF,03 07F
CFCF0430008
1310 DATA 80403C0302FA3641,1
476B35BA1D60E1F,307FFF7F7E61
E2A2,7FFFFFFFEFC7CE9,C6 018
0042040801C
1320 DATA 7F7F7F7F7F7F7FFF,F
FFFFFFFFFFFFFFFF,90E1C0E4F0F0
F8F9,8001010101818143,BF 5F5
771C8852244
1330 DATA C4C4E0E1D11A1E0E,1
FOFOF1F1F9F9FBF,E0808,FFBFBF
7FFFFFFFFF,F9FAFAFDFFEFF FFF
1340 DATA 4223A3D32205C9E8,8
80C8626B5DD244C,0A170B13A441
864,FEFEFEFEFE3E7F98,000 000
0000010181
1350 DATA 0000000000000000,0
0
1360 CALL CHAR(128,"19480007
97670598949141387A0900960189
747900080910080130750020 304
0504041")
1370 CALL CHAR(132,"00016700
A056869890198481714148174717
4009759600600589A0D09004 894
8000108")
1380 CALL HCHAR(9,13,88,8)::
B=12 :: FOR A=137 TO 143 ::
B=B+1 :: CALL HCHAR(10, B,A)
:: NEXT A :: CALL HCHAR(10,2
0,40)
1390 CALL CHAR(89,"FFFFFFFFF
FFFFFFFF"):: CALL HCHAR(1,1,8
9,256):: CALL HCHAR(17,1 ,89
,256):: CALL VCHAR(1,1,89,28
8):: CALL VCHAR(1,21,89,288)
1400 B=12 :: FOR A=41 TO 48
:: B=B+1 :: CALL HCHAR(11,B,
A):: NEXT A
1440 B=12 :: FOR A=73 TO 80
:: B=B+1 :: CALL HCHAR(15,B,
A):: NEXT A

```

```

1450 CALL HCHAR(16,13,81)::
CALL HCHAR(16,14,74):: B=14
:: FOR A=82 TO 87 :: B=B +1
:: CALL HCHAR(16,B,A):: NEXT
A
1455 !@P+
1460 SUBEND
1470 SUB BEAMMEUP
1474 GOTO 1480
1475 A
1477 CALL COLOR :: CALL SOUN
D :: CALL SPRITE :: CALL MAG
NIFY :: CALL DELSPRITE
1478 !@P-
1480 CALL COLOR(8,2,2):: GOT
O 1540
1490 FOR A=1 TO 2 :: CALL SO
UND(50,8000,26,12000,26)
1500 CALL SPRITE(#1,128,15,9
8,97,#2,132,15,98,118,#3,132
,15,65,97,#4,128,15,65,1 28)
1510 CALL SOUND(50,8000*1.00
5,18,12000*1.005,18)
1520 CALL SPRITE(#1,132,14,9
8,97,#2,128,14,98,128,#3,128
,14,65,97,#4,132,14,65,1 28)
1530 NEXT A :: RETURN
1540 CALL MAGNIFY(4):: GOSUB
1490 :: FOR A=2 TO 7 :: CALL
COLOR(A,10,2):: NEXT A
1550 CALL COLOR(14,10,2):: G
OSUB 1490 :: FOR A=2 TO 7 ::
CALL COLOR(A,6,2):: NEX T A
:: CALL COLOR(14,6,2):: GOSU
B 1490
1560 FOR A=2 TO 7 :: CALL CO
LOR(A,5,16):: NEXT A :: CALL
COLOR(8,2,16,14,5,16):: GOSU
B 1490 :: CALL DELSPRITE(ALL
)
1565 !@P+
1566 SUBEND
1570 SUB BEAMDOWN
1574 GOTO 1580
1575 A
1577 CALL COLOR :: CALL DELS
PRITE :: CALL SOUND :: CALL
SPRITE
1578 !@P-
1580 GOSUB 1610 :: FOR A=2 T
O 7 :: CALL COLOR(A,6,2):: N
EXT A :: CALL COLOR(14,6 ,2,
8,2,2):: GOSUB 1610
1590 FOR A=2 TO 7 :: CALL CO
LOR(A,10,2):: NEXT A :: CALL
COLOR(14,10,2):: GOSUB 1610
1600 FOR A=0 TO 14 :: CALL C
OLOR(A,2,2):: NEXT A :: GOSU
B 1610 :: CALL DELSPRITE (AL
L):: GOTO 1660
1610 FOR A=1 TO 2 :: CALL SO
UND(50,8000,26,12000,26)
1620 CALL SPRITE(#1,128,15,9
8,97,#2,132,15,98,118,#3,132
,15,65,97,#4,128,15,65,1 28)
1630 CALL SOUND(50,8000*1.00
5,18,12000*1.005,18)
1640 CALL SPRITE(#1,132,14,9
8,97,#2,128,14,98,128,#3,128
,14,65,97,#4,132,14,65,1 28)

```

```

NIFY :: CALL DELSPRITE
1478 !@P-
1480 CALL COLOR(8,2,2):: GOT
O 1540
1490 FOR A=1 TO 2 :: CALL SO
UND(50,8000,26,12000,26)
1500 CALL SPRITE(#1,128,15,9
8,97,#2,132,15,98,118,#3,132
,15,65,97,#4,128,15,65,1 28)
1510 CALL SOUND(50,8000*1.00
5,18,12000*1.005,18)
1520 CALL SPRITE(#1,132,14,9
8,97,#2,128,14,98,128,#3,128
,14,65,97,#4,132,14,65,1 28)
1530 NEXT A :: RETURN
1540 CALL MAGNIFY(4):: GOSUB
1490 :: FOR A=2 TO 7 :: CALL
COLOR(A,10,2):: NEXT A
1550 CALL COLOR(14,10,2):: G
OSUB 1490 :: FOR A=2 TO 7 ::
CALL COLOR(A,6,2):: NEX T A
:: CALL COLOR(14,6,2):: GOSU
B 1490
1560 FOR A=2 TO 7 :: CALL CO
LOR(A,5,16):: NEXT A :: CALL
COLOR(8,2,16,14,5,16):: GOSU
B 1490 :: CALL DELSPRITE(ALL
)
1565 !@P+
1566 SUBEND
1570 SUB BEAMDOWN
1574 GOTO 1580
1575 A
1577 CALL COLOR :: CALL DELS
PRITE :: CALL SOUND :: CALL
SPRITE
1578 !@P-
1580 GOSUB 1610 :: FOR A=2 T
O 7 :: CALL COLOR(A,6,2):: N
EXT A :: CALL COLOR(14,6 ,2,
8,2,2):: GOSUB 1610
1590 FOR A=2 TO 7 :: CALL CO
LOR(A,10,2):: NEXT A :: CALL
COLOR(14,10,2):: GOSUB 1610
1600 FOR A=0 TO 14 :: CALL C
OLOR(A,2,2):: NEXT A :: GOSU
B 1610 :: CALL DELSPRITE (AL
L):: GOTO 1660
1610 FOR A=1 TO 2 :: CALL SO
UND(50,8000,26,12000,26)
1620 CALL SPRITE(#1,128,15,9
8,97,#2,132,15,98,118,#3,132
,15,65,97,#4,128,15,65,1 28)
1630 CALL SOUND(50,8000*1.00
5,18,12000*1.005,18)
1640 CALL SPRITE(#1,132,14,9
8,97,#2,128,14,98,128,#3,128
,14,65,97,#4,132,14,65,1 28)

```

```

1650 NEXT A :: RETURN
1655 !@P+
1660 SUBEND

```

✎ END OF ARTICLE

LEARN TO KNOW YOUR TI

LESSON 24

with Percy Harrison

First, I must apologise for getting ahead of the lessons that I have given you up until this month as I have jumped the gun in using the GOSUB command in the answers to last months assignments which are included at the end of this lesson. In order to put things right, and so that you can better understand why I have used the GOSUB command, this lesson will explain to you the use of the GOSUB command in writing programs.

Like GOTO, GOSUB causes a jump to another line number. The only difference is that in GOSUB the computer stores the next line number following the GOSUB command on a stack. When the computer encounters a RETURN statement, it pops the line number off the stack and returns control to that line.

Subroutine calls can be nested at least 9 deep.

The END command can be put anywhere in the program and you can use as many end statements as you wish. All that END does is to return control to the command mode.

Subroutines are useful not only in long programs but in short ones where "chunking" the task into sections leads to clarity.

GOSUB was put into BASIC for making modules. This lesson shows modular construction in a graphics program. The same subroutine which writes the letter "J", in the example that we will use, also erases it.

The JUMPING J exercise will allow you to try many different effects in the moving graphics display.

LESSON 24 PRETTY PROGRAMS, GOSUB, RETURN

Run this program then save it to tape or disk:

```

100 REM MAIN PROGRAM
101 REM
110 PRINT "HOP TO THE SUBROUTINE"
120 GOSUB 200

```

```

130 PRINT "BACK FROM SUBROUTINE"
133 PRINT
135 PRINT "HOP AGAIN"
140 GOSUB 200
150 PRINT "BACK AGAIN"
190 END
200 REM SUBROUTINE
201 REM
210 PRINT "IN THE SUBROUTINE"
212 CALL SOUND(500,300,1)
215 FOR T=1 TO 1000
216 NEXT T
220 PRINT "PACK YOUR BAGS, BACK WE GO"
290 RETURN

```

This is the skeleton of a long program. The main program starts at line 100 and ends at line 190.

Where there are PRINT commands, you may put in many more program lines.

Line 120 and line 140 "call the subroutine". This means the computer goes to the lines in the subroutine, does them, and then comes back.

The GOSUB 200 command is like a GOTO 200 command except that the computer remembers where it came from so that it can go back there as soon as the subroutine is executed.

The RETURN command tells the computer to go back to the next statement after the GOSUB statement that activated it.

WHAT GOOD IS A SUBROUTINE

In a short program, not much.

In a long program, it does two things:

1. It saves you work. It saves space in memory. You do not have to type in the same program lines in different parts of the program.
2. It makes the program easier to understand and faster to write and debug.

MOVING PICTURES

Type in the following "JUMPING J" program, save it to tape or disk and then run it:

```

10 REM *** JUMPING J ***
20 CALL CLEAR
22 X=15
23 Y=12

```

```

24 D=1
25 FOR J=1 TO 5
26 FOR I=1 TO 2
30 CH=42
31 GOSUB 100
40 CH=32
41 GOSUB 100
45 Y=Y-D
50 NEXT I
55 D=-D
60 NEXT J
90 END
100 REM
101 REM === DRAW THE J ===
102 REM
110 CALL HCHAR(Y,X,CH,5)
120 CALL VCHAR(Y+1,X+2,CH,7)
130 CALL HCHAR(Y+7,X,CH,2)
140 CALL HCHAR(Y+6,X,CH)
190 RETURN

```

The picture is the letter "J". The subroutine starting in line 100 draws the "J". Before you GOSUB 100 you pick what character you want the "J" to be. Look at line 30 and at line 40. If you pick the space character, then the subroutine erases a "J" from that spot.

The subroutine draws the "J" with its upper left corner at the spot X,Y on the screen. When you change X or Y (or both) the "J" will be drawn in a different spot. Lines 22 and 23 say that the first "J" will be drawn near the middle of the screen.

The variable "D" tells how far the "J" will move from one drawing to the next. Line 24 makes "D" equal to 1, but line 55 changes "D" to -1 after 5 pictures have been drawn. A negative "D" makes the picture move down.

Line 45 says that each picture will be drawn at the spot where Y is smaller than the last Y by the amount D.

EXERCISE

Enter the JUMPING J program and run it. Then make these changes:

1. Change the subroutine so it prints your own initial.
2. Change the character to a solid square. Make its colour blue.
3. Change the "jumping" to "sliding" (so the initial moves horizontally instead of vertically).

4. Change the starting point to the lower right hand corner instead of the middle of the screen.
5. Change the distance the slide goes to 10 steps instead of 5.
6. Change the size of each step from 1 to 2.
7. Change the "sliding" so it slides uphill. Use

X=X+D

Y=Y-D

8. Change the program so that the initial changes colour from green (colour 3) through all the colours to white (colour 16) as it jumps.

Assignment 24

Write a short program which uses subroutines. It doesn't have to do anything useful, just print some silly things. In it put three subroutines:

Call one of them twice from the main program.

Call one of them from another of the subroutines.

ANSWERS TO LESSON 23

Assignment Question 23-1

```

10 REM MENU MAKER
12 CALL CLEAR
20 PRINT "WHICH COLOUR DO YOU LIKE"
21 PRINT
22 PRINT "<Y> YELLOW"
23 PRINT "<R> RED"
25 PRINT "<B> BLUE"
26 PRINT
30 CALL KEY(O,C,S)
31 IF C=-1 THEN 30
32 C$=CHR$(C)
35 IF C$<>"Y" THEN 40
36 C=12
37 GOTO 80
40 IF C$<>"R" THEN 45
41 C=7
42 GOTO 80
45 C=5
80 CALL SCREEN(C)
90 FOR T=1 TO 500
91 NEXT T

```

from Geoff Trott

```

10 REM SILLY SENTENCES
12 CALL CLEAR
13 PRINT"SILLY SENTENCES"
14 PRINT
15 PRINT "WANT INSTRUCTIONS <Y/N>"
16 PRINT
18 GOSUB 200
20 IF Y$="Y" THEN 100
21 PRINT"ENTER THE SUBJECT: (END WITH A PERIOD)"
22 PRINT
23 GOSUB 300
33 PRINT "ENTER THE VERB: (END WITH A PERIOD)"
34 PRINT
40 GOSUB 300
50 PRINT"ENTER THE OBJECT: (END WITH A PERIOD)"
51 PRINT
52 GOSUB 300
85 PRINT S$
99 END
100 CALL CLEAR
110 PRINT"THREE PLAYERS ENTER PARTS OF A SENTENCE"
115 PRINT"NO PLAYER CAN SEE WHAT THE OTHERS ENTER"
120 PRINT"THE FIRST ENTERS THE SUBJECT"
121 PRINT"(THE PERSON DOING SOMETHING)"
125 PRINT"THE SECOND ENTERS THE VERB"
126 PRINT"(THE ACTION WORD)"
130 PRINT"THE THIRD ENTERS THE OBJECT"
131 PRINT" (THE PERSON OR THING TO WHOM"
132 PRINT"THE ACTION IS DONE)"
150 FOR T=1 TO 2000
151 NEXT T
152 PRINT
199 GOTO 21
200 REM LOOK AT KEYBOARD
210 CALL KEY(O,Y,S)
220 IF Y=-1 THEN 210
230 Y$=CHR$(Y)
240 CALL KEY(O,Y,S)
250 IF S<>0 THEN 240
299 RETURN
300 REM GET A WORD
310 GOSUB 200
320 IF Y$="." THEN 390
330 S$=S$ & Y$
340 GOTO 310
390 S$=S$ & " "
399 RETURN

```

Remember when you type in the information requested by the program it will not show on the screen and if you don't type in a period after each entry the program will not progress to the next step.

Bye for now.

END OF ARTICLE

Letter from Pierre Garoche

Towards the end of last year, I received a letter from Pierre Garoche, who is our only member in France. Pierre often writes letters to me about some of my articles and always adds much to what I have said. This time he also had reason to change a Load and Run (D/F 80) file into a memory image (Prog) file for a game. His letter is as follows.

10th November 1994

Dear Geoff

Your article in the November issue of TND came at the right time. Last week, I have built a memory image from a large program named ATC. Your son may take an interest in ATC because ATC is a game. That makes two reasons to say that you had a good idea to issue your Techo-Time in TND #10.

I received ATC from Ian J. HOWLE, 3707 S.W. Southern St., SEATTLE, WA. 98126 (206) 938-4065 in thanks for some information I send him last year. In MICROpendium "reader to reader", Ian asked about bit map mode in assembler. A not very good answer was given in MICROpendium July issue, so I sent more information to Ian. This year, in September, Ian send me the ATC program in thanks for my help (he says). He also says that he is not able to upload the ATC file via modem. He is wondering if I could spread ATC around to any friend or BBS. I am not linked to a BBS but I have friends at TISHUG and I shall send a copy of ATC both to Larry Saunders and Percy Harrison.

Now, it is time to tell you how I managed to build a memory image file from ATC.

Before I decided to build a memory image file from the ATC file I know that ATC:

- is a load and run compressed object code program.
- is not a self starting program, its start name is START.
- cannot be loaded from Funnelweb loaders, it overwrites Funnelweb code.
- does not ruin the Funnelweb mail box.

As I do not have as much skill as you have to read tags in the code of a Load and Run file, I think it is better for me to analyse the CPU memory when the program ATC is loaded.

- The starting address will be found from the REF/DEF table.
- The memory addresses occupied by ATC code will be found if I clear CPU high memory before loading ATC, so the ATC code can be detected easily.

- Inspect ATC code to detect use of TI utilities and if workspace setting is done before any modification of registers from the start address.

From this information, I can build a specific program to save the occupied memory in suitable files. Each one of these files will have a bit of room in front of the code. Next this bit of room will be patched, using a sector editor, to turn each file into a part of the wanted memory image.

The file MI_MAKER gives in step by step form, the process I used. SAVLR/O is the specific tool. On the floppy you will find also some utilities. If you are of the opinion that some utilities among these may be of interest to TISHUG members I beg you to advise the shop.

Best regards.

As usual, Pierre has approached the problem from a programmer's point of view and written some assembler language programs to help him solve the problem. Here is the process he used.

==> MI_MAKER Pierre Garoche, Nov 1994

Last month I received a floppy disk loaded with a file named ATC. It is a game that must be loaded using the Editor Assembler loader option 3. The program does not start automatically and its start name (START) must be typed in.

The Editor Assembler loader option 3 spends a lot of time loading the file ATC, because ATC is a long file (171 sectors). A Load and Run loader is not a fast tool. It does a complex conversion of the code on the disk to load into memory. A memory image will be more convenient in the present case, as it is the fastest format to load a program and it is self starting. A memory image loader (Editor Assembler option 5) does not do any conversion, only a move from the disk to memory.

The problem springs from the large part of high memory where the program is stored. The tools I could use to build a memory image from the loaded ATC program overwrite a part of the memory where ATC is stored. I had to build special tools: they are named SAVLR/O, MARKEM/O. I take the occasion to explain the method step by step.

Short recall

1/ What is a memory image?

A memory image is a file or a bunch of files where the code of a program is stored exactly as it is when stored in memory; a memory image is loaded using the Editor Assembler loader option 5.

2/ Memory image format.

Each file of the bunch (or the only file) begins with 3 words in front of the program code. The 3 word block is named the HEADER; its 3 words are used by the loader to store in the right place in memory the code that is in the file after the header.

- The third word contains the first memory address into which the code must be loaded in the CPU memory.
- The second word contains the number of bytes to load.
- The first word is a flag that means: a zero value means this is last file of the memory image code; any other value means there is at least one more file to load. The non-zero value is usually >FFFF.

Each file of a memory image must not be longer than 8K (>2000 bytes = 6 bytes for the HEADER and >1FFA bytes for the code). (It can be longer than this but this is the normal maximum value generated by the SAVE programs.) If the program uses more than >1FFA consecutive bytes the memory image is a bunch of files, with the name of each file the same except for the last character whose ASCII code is incremented in each file successively.

3/ Memory image loaders.

A memory image loader ends its work by setting the Program Counter of the micro processor at the first address of the program. As a result this address must be the start point of the program. A memory image loader sets the workspace at >83E0 (GPL workspace). Among the various memory TI image loaders, the TI option 3 loaders do not load the TI utilities into low memory. It is the same for the Horizon RAMdisk loader, unlike Funnelweb, which brings in the TI utilities when the E/A loader is selected, and it is the same for Maximem.

The process

To build a memory image it is necessary to save the code of the program when it has been loaded into memory. Then the file header will be added to each file. Eventually the first file will be modified so the memory image loaders links the start address of the program and sometime the user workspace will be set by the modifications.

The method

How to build a memory image from the ATC file?

- 1/ Do an analysis of ATC code implantation in the memory.
- 2/ Save the ATC code and occasionally other code used by the program (the TI utilities) in suitable files.

3/ Add to each file the appropriate header.

Analysis of ATC code implantation.

- a/ At first clear high memory from >A000 to >FFFB using the debugger tool (DEBUG>6000 or DEBUG>2676).
- b/ Next load the ATC program but do not run it.
- c/ Then reload the debugger to analyse both code implantation and REF/DEF table.

The three steps a, b, c, must be done at a stretch from the Editor Assembler menu screen, using loader option 5 for DEBUG and loader option 3 for ATC. To clear high memory, clear the address >A000 using the M command. Next copy the cleared byte from >A001 to >FFFB using the N command. Also you may clear low memory from >3FF0 to >3FFF, reading the REF/DEF table will be more easy.

Now return to the Editor Assembler menu using the Q command. From this screen, load ATC, do not type the start name but press FCTN<9> (BACK) to return to the Editor Assembler screen and reload DEBUG. Now it is not difficult to find the memory field where the ATC program is stored and the address pointed to by the label START from the REF/DEF table.

Results:

ATC code starts from address >A04C to the address >FFBB and the start address of the program is at address >A050. The start point of the program is not at the beginning of the code. At address >A056 to >A05C is stored 0200 01E3 0420 211C. This is the code of the 2 instructions LI R0,>01E3: BLWP @VWTR. The program uses the workspace set by the loader and also uses the TI utilities. This information will have to be taken into consideration this when the memory image files are built.

Code modification:

When loaded, a Load and Run program starts using the user workspace (20BA to >20CF). It is different for a memory image program that starts using the GPL workspace (>83E0 to >83FF).

Something must be done to have the right workspace used and to start the program at the correct starting address. The code in the first memory image file must begin with:

```
LWPI >20BA 02E0 20BA
JMP >A050 1002
```

This adds 6 more bytes and it must be placed just in front of the original ATC code.

Saving the code

Each file must have a header and save the program code so each byte of the code will load into the right memory addresses. The memory contents must be saved in

files, with the first 6 bytes of each file not ending up as part of the actual memory image. This may be illustrated as follows:

```
file # n === .....
file # n+1 ===.....
(=== are the 3 words that will become file header)
(Program code .....)
```

The last file of the bunch is devoted to the TI utilities. As I am not a retailer, I put in the last file the low memory code from >2000 to >2675 as it is stored in memory by the Load and Run loader.

Next, after the bunch of files are saved, using a sector editor, the first 3 words in each file will be modified to become a header. In the first file the supplementary code will be added in the same manner, so the first file must start 12 bytes before the unmodified ATC code.

To save all the files I have written a little program: it is stored in low memory from address >2676 so it does not overwrite any part of the memory to be saved. The name of this tool is SAVLR/O. It returns in one shot four files: CODEPART1, CODEPART2, CODEPART3, CODEPART4. As I am wary of my new tools, I decided to do a preliminary test.

Loading high memory in such a way that each word contains its address, seemed to me a fine solution to test the SAVLR/O tool. This is done by loading at first the program MARKHM/O followed by the program SAVLR/O. Take my advice and do the same operations. Then using a sector editor, a sheet of paper and a pencil look at the files CODEPART1, CODEPART2 and CODEPART3.

Copy the first 10 words of CODEPART1 and the last word of the file. Copy the first 4 words of CODEPART2 and its last word. Copy the first 4 words of CODEPART3 and find the last saved address in the last sector of the file (the last word that is not zero). From these values, check that the memory is saved correctly with the required overlap for the header.

Now thinking about the modifications that will be done when the tool SAVLR/O will be used with a high memory ATC filled, verify that the end of CODEPART1 fits well with the start of code in CODEPART2 when the header will be set. Is it the same for CODEPART2/CODEPART3?

Is the last word of CODEPART3 the last address of ATC file? From all the information you have, try to find the correct header for each memory image file.

I make you see the returned files clearly; now, it is time to do the main work:

Load ATC using the Editor Assembler loader option 3, do not run the program, return to the Editor Assembler menu and load the tool SAVLR/O using the same loader. Answer the prompt and you will find 4 files CODEPART1, to CODEPART4. Copy each one using another name, ATC1, ATC2, ATC3, ATC4 and do the modifications in

the renamed files. These modifications are:

In ATC1 the 6 first words will be turned into:
FFFF 1FFA A046 02E0 20BA 1002, which is followed by
the ATC code: 02E0 A00C
In ATC2 the 3 first words will be turned into:
FFFF 1FFA C040, which is followed by the ATC code:
0589 0289 0003
In ATC3 the first 3 words will be turned into:
FFFF 1F82 E03A, which is followed by the ATC code:
80C0 0000 0001
In ATC4 the first 3 words will be turned into:
0000 0676 2000, which is followed by the low memory
code: A55A 2128 2398

The proof

From the Editor Assembler memory image loader
(option5) load ATC1. Unlock the alpha lock and have fun
using joystick #1.

Note: The ATC program, and ATC1/2/3/4 did not run
well in my system when my RAMdisks are set to power-up
ON.

To call the Editor Assembler menu, I power ON the
console pressing the shift key to stop the reset at the
TI MENU. Calling the Editor Assembler menu from the
HORIZON menu (pressing C key) brings a misfunction. I
must say that I have not the Editor Assembler module but
MAXIMEM which emulates the Editor Assembler module, but
perhaps it gives not a total emulation.

That is the end of Pierre's article. I hope it
provided another interesting way of converting display
fixed 80 program files into program or memory image
files. I found it very interesting that he worried
about which workspace was to be used. If you remember
your assembler language, the registers used by the
processor are located in memory according to the
contents of the workspace register. If the program is
written correctly, it should not matter where in memory
the registers are located, as long as that part of
memory is not used by the program. The only other
reason for the actual workspace pointer being important
would be that some of its registers were assumed to
already have certain values in them. This would be a
bad way of programming and would certainly lead to
problems if the wrong workspace were used. This may be
the case here as the ACT program did not run correctly
when I ran it from the ROOT menu program.

Pierre has provided some useful utility programs,
including several program versions of Debug which load
into >2676 (low memory), >6000 (cartridge space), >A050
(high memory above Funnelweb mailbox), and >E000
(towards the top of high memory). He also provided an
Editor Assembler loader from Barry Boone with its
Extended BASIC loader and the other two programs he

mentioned, SAVLR/O and MARKEM/O. If you are interested
in these programs, contact the shop.

I also had an interesting letter from Jim Banfield
from Armidale. He contributed a series of articles a
few years ago and uses his TI99/4A in a most unique way.
He is building a long word-length computer and uses the
TI99/4A as the input and output for this computer. He
observed that not many of our members appeared to be
interested in hardware, judging from my recent articles.
I think that many may feel they would like to know more
about the hardware but are unable to come to terms with
all the detailed knowledge that is required. Jim has
built his own disk controller and written the software
for its DSR. He then had trouble when he tried to put
all the disk controller programs in an EPROM at address
>4000. It worked fine at other addresses but not there.
The address range >4000 to >5FFF is shared by all the
I/O devices attached to the TI99/4A. A particular
device's programs are switched into that memory space by
using the CRU I/O of the processor. CRU uses only the
address lines for output and its own special line for
input to the processor. The only reason for problems in
using the address range would be that another device
like the RS232 card would be enabled at the same time.
The operating system checks all possible CRU addresses
whenever some I/O is done until it finds the one it
wants.

Meanwhile, I have been looking at the best GIF
viewing program I have come across. It was written by
Ton Brouwer of The Netherlands and is the easiest to use
I have seen. All these programs display pictures which
have a maximum of 256 colours and do not do a good job
with small changes in colour as with skin tones. They
all seem to be written for the 9938 video chip while we
now have the 9958 video chip. One of the enhancements
of the 9958 is a capability of the simultaneous display
of 19268 colours so it should be possible to get better
colours with the 9958 chip. My approach was to
disassemble the program and try to understand how it
works as I am not sure how GIF files store the data. I
do know that a GIF file on a PC gives better colour than
the same GIF file on the TI99/4A. I am some way down
the track in my quest, but I will leave this for another
article.

END OF ARTICLE



EXPANDING A TI99/4A for RS232/5 and 6.

by Ross Mudie, 3rd October 1994.

The Local Area Network (LAN) which I set up for the Shahzada Endurance Horse Ride event, (see article in TND October 1994), needed just one more terminal to meet the operational requirements of the event. The Server for the LAN was a TI99/4A with 2 x RS232 cards which allowed four terminals to be connected. The original design by Texas Instruments for the TI99/4A computer was for four to be the maximum number of RS232 ports. To connect extra terminals on the LAN, a non-standard approach has to be adopted to obtain extra serial ports. I am planning to use the same system again at the event next year with an extra terminal and some improvements in the programs.

Three methods were considered to achieve the extra ports:

- a) Modify the CRU address and ROM (using an EPROM) in a TI RS232 card.
- b) Modify the CRU address and EPROM in an AT Multi-Function Card (AT MFC).
- c) A hardware logic device to share one serial port over two terminals.

The last idea was least attractive because of the amount of hardware development which would be involved. Direct connection of each terminal to the server computer allows all the control work to be done in software. My how the thinking of this once "hardware junkie" has changed!

TI RS232 cards are not easy to obtain and modification of the CRU address would probably involve a modified PAL chip. Not a very attractive option.

To use the AT MFC would require component population of the RS232 parts of the AT MFC printed circuit board which already existed in the server computer, providing the Double Sided Double Density disk controller. After a short discussion with Geoff Trott, the expansion of the AT MFC was decided upon. Geoff offered to burn an EPROM to include RS232/5 and RS232/6 in the device names table in place of RS232/1 and RS232/2. The information which Geoff had available, showed that the CRU address for the AT MFC RS232 could easily be changed from the normal >1300 to >1B00 by the inclusion of just one inverter in address line A4 feeding to pin 4 of IC U8. This also placed the parallel port of the AT MFC at CRU address 1C00. There were already 2 spare gates available in a 74LS00 which was previously used to overcome a problem in the disk controller of the AT MFC.

Just one of these spare gates could be used to provide the inverter.

I decided to include a parallel port and 32K memory expansion on the AT MFC since the number of spare slots in the server PE box is becoming rather tight. Percy Harrison from the club shop arranged for a kit of parts for the expansion project.

After moving the 32K memory function into the AT MFC (which made a second PE box slot spare and provided a 32K memory card for use with another terminal), the allocated usage of slots in the server PE box is as follows:

SLOT	CARD	FUNCTION	CRU ADDRESS
1.	Bus expansion card.		
2.	TI Serial card -RS232/ 1 and 2.	-Parallel no 1 (PIO/1).	>1300
3.	TI Serial card -RS232/ 3 and 4.	-Parallel no 2 (PIO/2).	>1500
4.	RAMDISK.	Data storage	>1000
5.	CorComp Triple Tech	-Time of day clock.	>1D00
		-64K Printer buffer.	
6.	Spare.		
7.	AT MFC-Disk Controller.		>1100
		-32 K Memory.	
		-Serial ports RS232/ 5 and 6.	>1B00
		-Parallel no 3. (PRINT)	>1C00
8.	Spare.		

Once the component placement and soldering was complete, it was time to test out the expanded AT MFC. Initially all appeared to work well. Using my DSR PEEKER program, I found that the serial port of the AT MFC EPROM contained the names MIDI, MIDI2, RTTY, RTTY2, VIATEL, VIATEL2, RS232, RS232/5 and RS232/6 at CRU address >1B00. At CRU >1C00, I found PIO and PRINT in the device names table. If I attempted to print on PIO, it was printed out by the PIO with the lowest card address, namely, the card with CRU address of >1300. In the multi device environment where there are 3 identical device names, the device which is found first in the CRU search gets the job. To access the parallel port on the card with the CRU address of >1500 its device name has to be PIO/2 and to get to the parallel port on the AT MFC its device name has to be the alternate name of PRINT. Likewise, just specifying RS232 will give the job to the card with the >1300 CRU address and the "/number" suffix must be used to access each RS232 serial port.

I found that the AT MFC worked just like the TI RS232 card when it came to the programming department. My program opens each file in extended basic, in a FOR NEXT

loop, (to save memory), as follows:

```
290 FOR S=1 TO 6
300 OPEN #S:"RS232/"&STR$(S)&".BA=9600.EC.LF",VARIABLE
      255
310 NEXT S
```

This opens 6 files, numbered 1 to 6, RS232/1 through to RS232/6. The availability of /5 and /6 in the RS232 part of the AT MFC EPROM made the job just that little bit easier. The reading from the serial port is performed in a linked assembly routine which scans the inputs. Due to the wired handshaking between the terminals and the server in the LAN, there is no problem with making any terminal wait until the server is ready to process the information. This very nicely dispenses with the need to use interrupts. The extended basic program operates with CALL FILES(9) which allows 9 disk files to be open simultaneously. There are 9 other non disk files which are permanently open.

The Carrier Detect (CD) and Data Terminal Ready (DTR) have not been provided for the second RS232 on the AT MFC. This was a hardware design rationalisation but only in the interface to the RS232 port. The DTR and CD are both available from the TMS9902 communications controller. To provide these hardware handshaking lines so that a possible sixth terminal can be used, I will be soldering a second MC145406 on top of U24 and the necessary wiring will be provided.

You will have noticed that a data rate of 9600 bauds is used between computers in the LAN, to get the maximum possible speed. This is where problems of data corruption occurred with the AT MFC due to a power supply problem. An article has been prepared on this subject. The problem of data corruption was easily fixed as will be found in the article.

Once again its quite amazing what can be done with the old TI99/4A at a reasonable cost with the support of members of the user group. My special thanks to Geoff Trott and Percy Harrison for their assistance with setting up RS232/5 and RS232/6 for my TI99/4A LAN.

 **END OF ARTICLE**

**JUST A ONELINER
(ED)**

- Q. Which hand should you use to stir tea?
A. Neither. It is better to use a spoon.

PUZZLE

This months list of words is based around the subject of "AS FRESH AS"

D Q F T N E C E R B I O C S D F Z C P Z
I T T I B K D T N E R E F F I D R N A B
V Q Z L N C E T H S S Y U Y Z I K L W O
T L G P O V K B Z U H G H F S Z H B Q E
U O M O M P W Y Z L N X L P Z T Q T Q Z
L D L V U G U A G G C N W T D Z R I M D
I A C S D N Z J P N U J Z C J A J W E A
N Q T E R A U T I N I N U E W E B F J C
Y R P E R E Y E E Z O H D Y J R G T L N
M D D W S U V G L V D E S J T S L E G H
F O N Q X T P H S P Y D F E S K A L W U
M J B I G Y C J E E C E F P R N D E F J
E N F F W H M C T A Q N A A Z F N E S E
F V K P I E L H G K L R O S K J E T L C
O X Q L E E G F S Q K T P V N B I R Q K
K K L V A I S I B L G O H U E F Z Y K W
Y Y S R R G R H I W Z C G Y F L L Z E Z
N C R B S B G N W P U W X X P W N H P T
L T B A C D G B I Y C T Y E Z B V C O V
Z P P I N B Y U I O C J Z B L V O S U U

Find these hidden words

HOW TO PLAY

In this puzzle there are (20) words somewhere, horizontally, vertically, diagonally even backwards.

GOOD LUCK

BRIGHTEYED	BRISK	CHILLY
CLEAN	CLEAR	COOL
CRISP	DIFFERENT	HEALTHY
LATEST	MODERN	NEW
NOVEL	PURE	RAW
RECENT	REFRESHING	SPARKLING
STIFF	WINDY	

This puzzle was compiled using Ashley Lynn's programme "word puzzle" which can be ordered through TIsHUG.



BASCON

from Stephen Shaw in England.

There are many ways of producing program listings, and many ways of manipulating programs. This little article is to show you one program, which has been manipulated with three others, so you will see the original and three variations. Not shown here is the 28 column format, produced by Tony McGovern's COLIST program, as that has been used many times in this magazine already.

COLIST, UNBASHER, and NEATLIST are all available from the User Group disk library, while SMASH is a commercial program available from Tenex.

First the original program- I am only showing a central portion of the program, rather than take up too much magazine space!

```
1440 FOR X1=1 TO 4
1460 R(X1)=ASC(SEG$(P$,X1,1))
1480 NEXT X1
1500 W=0
1520 FOR I=1 TO 4
1540 FOR J=1 TO 4
1560 IF G(I)<>R(J)THEN 1640
1580 W=W+1
1600 R(J)=0
1620 GOTO 1660
1640 NEXT J
1660 NEXT I
1680 W=W-B
1700 RETURN
```

Once this program has been SMASHed, these lines look like this:

```
1440 FOR X1=1 TO 4 :: R(X1)=ASC(SEG$(P$,X1,1)):: NEXT X1
:: W=0 :: FOR I=1 TO 4 :: FOR J=1 TO 4 :: IF
G(I)<>R(J)THEN 1640
1580 W=W+1 :: R(J)=0 :: GOTO 1660
1640 NEXT J
1660 NEXT I :: W=W-B :: RETURN
```

Which can be made a little more meaningful by running it through NEATLISTER:

```
01440 FOR X1 = 1 TO 4 ::
  R(X1) = ASC(SEG$(P$ , X1 , 1)) ::
  NEXT X1 ::
  W = 0 ::
  FOR I = 1 TO 4 ::
    FOR J = 1 TO 4 ::
      IF G(I) < > R(J) THEN 1640
01580 W = W + 1 ::
  R(J) = 0 ::
  GOTO 1660
```

```
01640 NEXT J
01660 NEXT I ::
      W = W - B ::
      RETURN
```

VARIABLES USED IN MAIN PROGRAM
(in lines 1440 to 1700 only)

G	01440
I	01440 01660
J	01440 01580 01640
P\$	01440
R	01440
W	01440
X1	01440

(The variable listing is available as an option with the listing, or on its own).

We can also resplit the SMASHed program by using UNBASHER, which produces the following- note that the output has not been resequenced, but of course it could have been:

```
1440 FOR X1=1 TO 4
1441 R(X1)=ASC(SEG$(P$,X1,1))
1442 NEXT X1
1443 W=0
1444 FOR I=1 TO 4
1445 FOR J=1 TO 4
1446 IF G(I)<>R(J)THEN 1640
1580 W=W+1
1581 R(J)=0
1582 GOTO 1660
1640 NEXT J
1660 NEXT I
1661 W=W-B
1662 RETURN
```

You may not have a lot of control over the format of a program you purchase or receive from elsewhere- such as the library- but the utilities are available to change the format to something which may be easier to read/debug, or which perhaps occupies a little less memory space and runs a trifle faster.

 **END OF ARTICLE**

LEXMARK PRINTER
By Larry Saunders

I have been using a IBM Excjet II inkjet printer for several months now, and I am getting go results from it using. TI-Artist, with the EPSOM-IBM emulation.

With Page Pro, Banner Maker, and Poster Maker, they will work excellent using, 24 Pin and High Res, but they will print only garbage if you don't use Hi Res.

One of the pictures printed on the IBM Excjet II, using a TI99/4A computer is the house on the cover of TiSHUG News Digest January/February 1995.

Programing The IBM Compatibles in Basic

by B.v.Takach

You have not seen my by-line in the TND for some time. This is not due to a chronic writer's cramp, nor have I abandoned the 99/4A. The reason is simply that I did not have a worthwhile topic. Only politicians babble on ad nauseam boring one to death when it would be prudent to remain silent.

Now I have something which may interest some of you. Some BASIC programing hints for the PC users. In fact this is my first program, which has not seen the light of day on the TI. It is a commercial program, which would be unsaleable in Extended Basic format. The listing is incomplete as the section executing the calculations and the printouts have been omitted for several reasons.

Firstly, as most of you believe die casting is a new form of illegal gambling, I will spare you the disappointment at the end of a lengthy typing session when some incomprehensible gibberish is displayed on your monitor.

Secondly, the program has a solid price tag on it. Remember the golden words of our erstwhile leader, "There is no such thing as a free lunch." You have to be satisfied with a free morning cup of tea!

Programming the PC is not quite the enjoyable experience one is used to with the TI. The missing DISPLAY AT and ACCEPT AT statements with all the fancy bells and whistles makes the task much more complicated - as you will notice in a few minutes.

The program starts at line 20000, the start of the subroutine line 50 has commanded. The subroutine ends with line 21300.

[include listing 1-50 and 20000-21300 here]

This segment will present the first two colourful screens. The first screen is the password input routine (lines 20000-21000). It will ask for the password (line 20500), if it is not kosher then you will have two more chances. You failed twice more? The routine will dump you and delete the program from memory. If it was saved in protected mode ("pgm name", p) then it cannot be saved or listed. This makes the program relatively secure, you really need John Paine on your side to crack it and change the password somehow. Naturally the password is case sensitive. It will reject upper case characters if the stored password is in lower case.

This subroutine may be used on its own or tacked on to any other program.

Naturally line 21000 has to be changed to suit. The second screen is just a message screen and a reminder to prepare the printer. Basic is very impatient when it comes to printout. A forgotten printer will result in a cryptic timeout error message and you will be unceremoniously tipped out of your program.

My habitual omission to load paper, press the "on line" key or to switch on the printer altogether, prompted me to include this screen, besides it looks very professional. Enthusiastic programmers could fill the empty centre field with some inspired graphics or text. Beware of porno/racist or sexist text to safeguard against any future court battle charged with some new fangled offence under the anti-discrimination act. Lines 21010 - 21200 may be omitted if not needed or if the program does not use the printer.

[include the first two screen dumps here]

The actual program starts with line 100. Lines 300 - 365 and subroutine 7300 - 7380 produce the 3rd screen page, which is the first actual program page. The input subroutine (7300 - 7380) is a simple Fred Flintstone inspired segment with a very limited safeguard against wrong entry, however for those who manage to make a boo-boo, a "wish to correct?" escape route is still open. This reflects my philosophy, you can make a mistake, but you do not have to be an idiot.

Your attention is called to line 7326; the input\$(n) function is not known to EX.BASIC of TI. It is a more foolproof method of data entry than INPUT, however it can also cause some confusion, as you can see in lines 7330 and 7335. The fancy footwork with -1 and +1 was necessary because of 0. VAL(n\$) returns 0 for any non digit string. The calculation section of the program calls for R12 to be between 0-3. It was easier to restore the 0-1-2-3 order here than edit some 50 odd lines elsewhere.

[include listing of lines 100 - 190, 300 - 365 and 7300 - 7380 here]

We have passed through a number of non-TI Basic features so far, but the real piece de resistance is in the data acquisition section of the program (lines 5000 - 6790). This almost 2 pages full of code is necessary to overcome the handicap of the non-existent DISPLAY AT and ACCEPT AT features of PC based BASICS.

The code is bullet proof, but apart from a few non-essential lines it cannot be abbreviated much. The lines which do not make sense are related to the calculation segments. One can wander up and down the screen using the up/down arrow keys, change the entered data at will and the screen is re presented again with the last entered data in the event of a rerun because the results are unsatisfactory. The screen dump of the last two screen dumps show the presentation of the data on the monitor.

[include the 2 screen dumps here]

The program is booted by a simple 2 line bat file (I named it BETA.BAT):
@ ECHO OFF
GWBASIC.EXE E.BAS

Summary

The presented program highlights the differences between TI-s EXTENDED BASIC and GW BASIC. GW BASIC lacks some important refinements of EXTENDED BASIC, however it also has a number of features not available to the users of EXTENDED BASIC.

In addition, GW BASIC has an added trap for the unwary. The program presented was written and used with GW BASIC Ver.2.01(1984). The screens will turn into a hopeless mess if it is booted through the 1988 vintage GW BASIC Ver.3.23. According to the user manuals of both versions, the program is fine. Apart from lines 20000 -21300, there are no compatibility problems.

So far I have been unable to find out the reason for the non compatibility, but other GW BASIC users have also confirmed that certain programs will only run properly with the version it was produced by.

You may encounter other frustrations during programming, when contrary to the user manual the PC refuses to obey your commands.

I had a long and loosing battle with GWBASIC's MERGE option. Saving the program in ASCII code with the ,A option would not work. At the point of accepting defeat I tried to list the program to a disc file. There is no mention of this option in the 400 pages GWBASIC user's manual!

Anyway this is how you can do it:
LIST line number-line number,"Drive:FILENAME"
for example LIST -500,"A:LISTING". Bingo,
you created an ASCII coded listing in spite
of the uncooperating GWBASIC.

By the way, there is no need to use any extension when saving a basic program, GWBASIC is not fussy, the default is automatic and is always .BAS.

```
1 REM SAVE"GRAVITY.BAS
50 GOSUB 20000
```

```
100 REM *** GRAVITY FEED SYSTEM PROGRAM; Copyright:
    BETA ***
120 DIM Q$(12),V(12),H$(5),W$(5),OBE$(12)
125 REM INPUT-ADATOK MEGNEVEZESEI
130 Q$(0)=" 0. SPRUE AND POURING CUP HEIGHT HA mm
    .....:"
135 Q$(1)=" 1. DISTANCE FROM CENTRE OF GATE TO TOP
    OF CAVITY mm.:"
140 Q$(2)=" 2. NET HEIGHT OF THE CASTING
    mm.....:"
145 Q$(3)=" 3. THINNEST SIGNIFICANT WALL THICKNESS
    mm.....:"
150 Q$(4)=" 4. NET CASTING WEIGHT
    g.....:"
155 Q$(5)=" 5. WEIGHT OF SPRUE AND POURING CUP
    g.....:"
160 Q$(6)=" 6. WEIGHT OF RUNNERS g.....
    .....:"
165 Q$(7)=" 7. WEIGHT OF RISERS (FEEDERS) g.....
    .....:"
170 Q$(8)=" 8. OTHER MASS PAST GATE NOT INCLUDED
    ABOVE g .....:"
175 REM Q$(9)=" 9. ....
    .....:"
180 REM Q$(10)="10. ....
    .....:"
190 REM Q$(11)="11. ....
    .....:"

300 H$(1)="DATE "
301 H$(2)="COMPONENT NAME OR ID. "
302 H$(3)="PART NUMBER "
303 H$(4)="CUSTOMER'S REF. "
310 H$(5)="ORDER NUMBER "
350 LOCATE 6,3:PRINT H$(1):LOCATE 6,37:PRINT
    DATUM$:W$(1)=DATUM$
360 FOR I=2 TO 5:LOCATE 5+I,3:PRINT H$(I):LOCATE
    5+I,37:INPUT"",W$(I):NEXT I
365 GOSUB 7300
370 G=4:P=0:K=8:GOSUB 7100:GOSUB 5000
```

```
5000 FOR I=P TO K
5010 IF VIZS=0 THEN OBE$(I)=" 0.00"
5020 LOCATE 5+I+G,3:PRINT Q$(I):NEXT I:
5030 COLOR 0,2:FOR I=P TO K :LOCATE 5+I+G,60:PRINT
    OBE$(I):
5040 NEXT I
5050 I=P:T=1:H1=2:Y=60
5060 FF$=OBE$(I):H=LEN(OBE$(I)):X=5+I+G:GOSUB 6000
5070 IF VV=1 THEN 5060 ELSE OBE$(I)=FF$
5080 IF VV=2 THEN IF I=P THEN BEEP:GOTO 5060 ELSE
    I=I-1:GOTO 5060
5090 IF VV=3 THEN IF I=K THEN BEEP:GOTO 5060 ELSE
    I=I+1:GOTO 5060
5100 IF VV=4 THEN LOCATE 22,3:COLOR 7,4:PRINT
    "INPUT ERROR! ":BEEP:FOR I=1 TO 5000:NEXT
    I:GOTO 5060
5110 V(I)=VAL(OBE$(I))
5120 IF V(I)<0 THEN BEEP:LOCATE 22,3:COLOR
    7,4:PRINT"INPUT ERROR! ";:FOR J=1 TO 2500
    :NEXT J:GOSUB 600:GOTO 5060
5130 IF I<K THEN I=I+1:GOTO 5060
5140 LOCATE 22,3:COLOR 7,4:PRINT "DO YOU WISH TO
    CORRECT ? (Y/N)";:GOSUB 7000:
5150 IF V$="Y" OR V$="y" THEN GOSUB 7100:GOTO 5050
    ELSE IF V$="N" OR V$="n" THEN GOSUB 7100:GOTO
    5170 ELSE BEEP:GOTO 5140
5160 IF V$="Y" OR V$="y" THEN 5050
5170 RETURN
6000 REM ### ADATBEVITEL ###
6010 IF X<1 OR X>25 THEN 6100
6020 IF Y<1 OR Y>80 THEN 6100
6030 IF T=3 THEN H=8:H1=0
6040 IF H=0 THEN 6100
6050 IF T=1 THEN IF H1<0 OR H1>=H-1 THEN 6100 ELSE
6070 6060 GOTO 6110
6070 IF H1=0 THEN T1=0 ELSE T1=H-H1
6080 IF T<1 OR T>3 THEN 6100
6090 GOTO 6110
6100 VV=4:GOTO 6790
6110 JJ=0:LOCATE X,Y,1:COLOR 0,3:PRINT FF$:LOCATE
    X,Y:L=1
6120 D$=INKEY$:IF D$="" THEN 6120
6130 IF ASC(D$)=0 THEN 6150
6140 IF D$=CHR$(27) THEN VV=1:GOTO 6790 ELSE 6290
6150 D=ASC(MID$(D$,2,1))
6160 IF D=71 THEN JJ=1:L=1:LOCATE X,Y:GOTO 6120
6170 IF D=72 THEN VV=2:FG$="FEL":GOTO 6280
6180 IF D=75 THEN IF L=1 THEN 6120 ELSE JJ=1:L=L-
    1:GOTO 6230
6190 IF D=77 THEN IF L=H THEN 6120 ELSE
    JJ=1:L=L+1:GOTO 6230
6200 IF D=79 THEN JJ=1:L=H:LOCATE X,Y+L-1:GOTO 6120
6210 IF D=80 THEN VV=3:FG$="LE":GOTO 6280
6220 BEEP:GOTO 6120
6230 IF T=3 THEN 6260
6240 IF L=T1 THEN IF D=77 THEN L=L+1 ELSE L=L-1
6250 LOCATE X,Y+L-1:GOTO 6120
6260 IF L=3 OR L=6 THEN IF D=77 THEN L=L+1 ELSE
    L=L-1 6270 GOTO 6250
6280 IF T=1 THEN 6580 ELSE 6790
6290 ON T GOTO 6300,5660,6700
6300 IF D$>="0" AND D$<="9" THEN 6340
6310 IF D$="." THEN 6490
6320 IF D$=CHR$(13) THEN 6570
6330 BEEP:GOTO 6120
6340 IF JJ=0 THEN FF$=SPACE$(H) ELSE 6370
6350 IF T1<>0 THEN MID$(FF$,T1,1)=". "
6360 LOCATE X,Y:PRINT FF$:LOCATE X,Y
6370 JJ=1:PRINT D$;:MID$(FF$,L,1)=D$:L=L+1:IF L=T1
    THEN L=L+1:PRINT ". ";
6380 IF L>H THEN L=H:BEEP:LOCATE X,Y+L-1
6390 GOTO 6120
6400 IF T1=0 THEN J2=H ELSE J2=T1-1
6410 J1=0:FOR UU=1 TO J2
6420 IF J1=0 THEN IF MID$(FF$,UU,1)="" THEN 6440
    ELSE J1=1:GOTO 6440
6430 IF MID$(FF$,UU,1)="" THEN 6450
6440 NEXT UU
6450 UU=UU-1:MID$(FF$,1,J2)=SPACE$(J2-
    UU)+LEFT$(FF$,UU):RETURN
6460 IF T1=0 THEN L=H:BEEP:GOTO 6430
6470 MID$(FF$,T1+1,H1)=STRING$(H1,"0"):L=T1+1
6480 LOCATE X,Y:PRINT FF$:RETURN
```

```

6490 IF JJ=0 THEN FF$=SPACE$(H):IF T1<>0 THEN
MID$(FF$,T1-1,H1+2)=STRING$(H1+2,"0"):
MID$(FF$,T1,1)=" ":LOCATE X,Y:PRINT
FF$:LOCATE X,Y:T1:=T1+1:JJ=1:GOTO 6120 6500
GOSUB 6590:GOSUB 6460:LOCATE X,Y+L-1:GOTO 6120
6510 GOSUB 6400:GOSUB 6480
6520 IF T1=0 THEN 6560
6530 GOTO 6540
6540 FOR UU=H TO T1+1 STEP -1:IF MID$(FF$,UU,1)="
"THEN MID$(FF$,UU,1)="0"
6550 NEXT UU:LOCATE X,Y:PRINT FF$
6560 RETURN
6570 GOSUB 6510:VV=0:GOTO 6790
6580 GOSUB 6510:GOTO 6790
6590 IF T1<>0 THEN IF L>T1 THEN L=T1
6600 J1=0:FOR UU=1 TO L-1
6610 IF J1=0 THEN IF MID$(FF$,UU,1)=" " THEN 6630
ELSE J1=1:GOTO 6630
6620 IF MID$(FF$,UU,1)=" " THEN 6640
6630 NEXT UU
6640 IF T1=0 THEN J2=H ELSE J2=T1-1
6650 UU=UU-1:MID$(FF$,1,J2)=SPACE$(J2-
UU)+LEFT$(FF$,UU):RETURN
6660 IF D$=CHR$(13) THEN VV=0:GOTO 6790
6670 PRINT D$:MID$(FF$,L,1)=D$
6680 IF L=H THEN BEEP:LOCATE X,Y+L-1 ELSE L=L+1
6690 GOTO 6120
6700 IF D$=CHR$(13) THEN VV=0:GOTO 6790
6710 IF D$<"0" OR D$>"9" THEN BEEP:GOTO 6120
6720 IF JJ=0 THEN FF$=SPACE$(8):JJ=1:LOCATE
X,Y:PRINT FF$:LOCATE X,Y
6730 PRINT D$:MID$(FF$,L,1)=D$
6740 IF L=H THEN BEEP:GOTO 6770
6750 L=L+1
6760 IF L=3 OR L=6 THEN L=L+1:LOCATE X,Y+L-1
6770 LOCATE X,Y+L-1
6780 GOTO 6120
6790 COLOR 7,0:RETURN

```

```

7000 V$=INKEY$:IF V$="" THEN 7000 ELSE RETURN
7100 LOCATE 22,3:COLOR 7,0:PRINT SPC(76):RETURN
7200 FOR I=4 TO 20 :LOCATE I,3:PRINT SPC(76):NEXT
I:RETURN

```

```

7300 LOCATE 13,3:PRINT "Select Alloy Type:":LOCATE
15,5:PRINT "AL. (AP201, AP501, [Si<1]) = 1"
7310 LOCATE 16,5:PRINT "AL. (AS303, [ Si 3.5-6%] )
= 2":LOCATE 17,5:PRINT "AL. (BP401, [Si 10-
13.5%]) = 3": LOCATE 18,5:PRINT "ALUMINIUM
BRONZE OR BRASS = 4"
7320 LOCATE 13,38:PRINT "Select Riser/Gating
Method ":LOCATE 15,40:PRINT "Top Gated Design
=1":LOCATE 16,40:PRINT "Side Gated Design
=2":LOCATE 17,40:PRINT "Bottom Gated Design
=3"
7325 LOCATE 18,40:PRINT "Flat Casting &
Horizontal Gate = 4"
7326 LOCATE 19,9:PRINT "Your choice":LOCATE
19,22:R1$=INPUT$(1):R1=VAL(R1$)
7327 IF R1<1 OR R1>4 THEN BEEP:GOTO 7326 ELSE
LOCATE 19,22:PRINT R1
7330 LOCATE 19,44:PRINT "Your
choice":LOCATE 19,64:R2$=INPUT$(1):R2=VAL
(R2$)-1
7335 IF R2<0 OR R2>3 THEN BEEP:GOTO 7330 ELSE
LOCATE 19,64:PRINT R2+1
7340 LOCATE 23,3:PRINT "DO YOU WISH TO CORRECT ?
(Y/N)":GOSUB 7000:
7350 IF V$="Y" OR V$="y" THEN LOCATE 22,3:PRINT
SPC(35):GOTO 350 ELSE IF V$="N" OR V$="n" THEN
7380 ELSE BEEP:GOTO 7340
7360 IF V$="Y" OR V$="y" THEN 350
7380 FOR I=6 TO 19:LOCATE I,3:PRINT SPC(74):NEXT
I:RETURN

```

```

20000 REM *** keret készítő *** SCREEN DISPLAY ***
SAVE: PWORD.BAS
20002 KEY OFF:COLOR 1,0
20005 CLS:DATUM$=MID$(DATE$,4,2)+"."+LEFT$(DATE$,2)
+"."+RIGHT$(DATE$,4)+" "
20010 PRINT " ";STRING$(78,196);" "
20020 PRINT " ";STRING$(78,32);" "
20022 PRINT " ";STRING$(78,32);" "
20024 PRINT " ";STRING$(78,32);" "
20030 PRINT " ";STRING$(78,205);" "
20040 FOR PCI=1 TO 15 : PRINT
" ";STRING$(78,32);" ":NEXT PCI
20050 PRINT " ";STRING$(78,205);" "
20060 PRINT " ";STRING$(78,32);" "
20070 PRINT " ";STRING$(78,196);" "
20071 LOCATE 2,2:COLOR 7,4:PRINT " ":LOCATE
3,2:COLOR 2,7:PRINT " ":LOCATE
4,2:COLOR 7,2:PRINT " ":
20080 LOCATE 2,23 :COLOR 0,2:PRINT " Gravity
die design program ":LOCATE 3,15:PRINT "
for the design of feed system and sprue.
":LOCATE 4,28:PRINT " ":
20085 LOCATE 2,70:COLOR 4,0:PRINT " BETA ":LOCATE
3,70:COLOR 7,0:PRINT "SYDNEY":LOCATE
4,68:COLOR 2,0:PRINT "DATUM$"
20090 COLOR 0,7:LOCATE 21,35:PRINT " Messages:
":COLOR 7,0
20095 COLOR 7,4:LOCATE 22,8:PRINT "Scroll: U/D
Arrows,Selection: RETURN Back to
home:Esc";:
20100 REM RETURN
20500 LOCATE 10,15:COLOR 14,0:PRINT "Type in
Password! ":LOCATE 10,35:A$=INPUT$(5)
20550 P$="dukai"
20560 IF A$<>P$ THEN PU=PU+1 ELSE 21000
20562 IF PU=3 THEN LOCATE 15,20:COLOR 14,1:PRINT
"Sorry this is not the password !
":LOCATE 17,30:PRINT "I will terminate the
run":FOR I=1 TO 15000:NEXT I:CLS:SYSTEM
20563 IF PU<>3 THEN LOCATE 15,15:COLOR 14,1:PRINT
"Wrong password ! You have ";3-PU;" more
chance ! ":FOR I=1 TO 15000:NEXT I:LOCATE
15,15:COLOR 7,0:PRINT SPC(50):GOTO 20500
21000 REM *** Program Start ***
21005 LOCATE 10,15 :PRINT SPC(18):LOCATE 8,8:PRINT
"CONSULT THE USER'S MANUAL TO DETERMINE:"
:LOCATE 10,10:PRINT "- Alloy Selection
Choice":LOCATE 11,10:PRINT "-Pouring
Methods":
21010 LOCATE 12,10 :PRINT "- Feed System & Casting
Geometry"
21020 COLOR 0,7:LOCATE 21,35:PRINT " MESSAGES:
":COLOR 7,0
21200 COLOR 7,4:LOCATE 22,3:PRINT "PLEASE SWITCH ON
YOUR PRINTER PRESS RETURN TO
CONTINUE ":GOSUB 7000:A=ASC(V$):IF A=13 THEN
GOSUB 7100:COLOR 7,0:GOTO 21300 ELSE
BEEP:GOTO 21000 21300 FOR I=3 TO 12:LOCATE
I,3:PRINT SPC(40):NEXT I:RETURN

```

Gravity die design program for the design of feed system and sprue.		BETA SYDNEY 15.11.1994.																		
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 80%;">0. SPRUE AND POURING CUP HEIGHT HA mm</td> <td style="width: 20%; text-align: right;">0.00</td> </tr> <tr> <td>1. DISTANCE FROM CENTRE OF GATE TO TOP OF CAVITY mm.:</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>2. NET HEIGHT OF THE CASTING mm.....:</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>3. THINNEST SIGNIFICANT WALL THICKNESS mm.....:</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>4. NET CASTING WEIGHT g.....:</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>5. WEIGHT OF SPRUE AND POURING CUP g</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>6. WEIGHT OF RUNNERS g</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>7. WEIGHT OF RISERS (FEEDERS) g</td> <td style="text-align: right;">0.00</td> </tr> <tr> <td>8. OTHER MASS PAST GATE NOT INCLUDED ABOVE g</td> <td style="text-align: right;">0.00</td> </tr> </table>			0. SPRUE AND POURING CUP HEIGHT HA mm	0.00	1. DISTANCE FROM CENTRE OF GATE TO TOP OF CAVITY mm.:	0.00	2. NET HEIGHT OF THE CASTING mm.....:	0.00	3. THINNEST SIGNIFICANT WALL THICKNESS mm.....:	0.00	4. NET CASTING WEIGHT g.....:	0.00	5. WEIGHT OF SPRUE AND POURING CUP g	0.00	6. WEIGHT OF RUNNERS g	0.00	7. WEIGHT OF RISERS (FEEDERS) g	0.00	8. OTHER MASS PAST GATE NOT INCLUDED ABOVE g	0.00
0. SPRUE AND POURING CUP HEIGHT HA mm	0.00																			
1. DISTANCE FROM CENTRE OF GATE TO TOP OF CAVITY mm.:	0.00																			
2. NET HEIGHT OF THE CASTING mm.....:	0.00																			
3. THINNEST SIGNIFICANT WALL THICKNESS mm.....:	0.00																			
4. NET CASTING WEIGHT g.....:	0.00																			
5. WEIGHT OF SPRUE AND POURING CUP g	0.00																			
6. WEIGHT OF RUNNERS g	0.00																			
7. WEIGHT OF RISERS (FEEDERS) g	0.00																			
8. OTHER MASS PAST GATE NOT INCLUDED ABOVE g	0.00																			
MESSAGES: _____																				

Gravity die design program for the design of feed system and sprue.		BETA SYDNEY 15.11.1994.												
DATE	15.11.1994.													
COMPONENT NAME OR ID.	Small Acme													
PART NUMBER	AS/009													
CUSTOMER'S REF.	Proj.9411													
ORDER NUMBER	E-567													
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 45%;">Select Alloy Type:</td> <td style="width: 55%;">Select Riser/Gating Method:</td> </tr> <tr> <td>AL. (AP201, AP501, [Si<1]) = 1</td> <td>Top Gated Design = 1</td> </tr> <tr> <td>AL. (AS303, [Si 3.5-6%]) = 2</td> <td>Side Gated Design = 2</td> </tr> <tr> <td>AL. (BP401, [Si 10-13.5%]) = 3</td> <td>Bottom Gated Design = 3</td> </tr> <tr> <td>ALUMINIUM BRONZE OR BRASS = 4</td> <td>Flat Casting & Horizontal Gate = 4</td> </tr> <tr> <td style="text-align: center;">Your choice 2</td> <td style="text-align: center;">Your choice 3</td> </tr> </table>			Select Alloy Type:	Select Riser/Gating Method:	AL. (AP201, AP501, [Si<1]) = 1	Top Gated Design = 1	AL. (AS303, [Si 3.5-6%]) = 2	Side Gated Design = 2	AL. (BP401, [Si 10-13.5%]) = 3	Bottom Gated Design = 3	ALUMINIUM BRONZE OR BRASS = 4	Flat Casting & Horizontal Gate = 4	Your choice 2	Your choice 3
Select Alloy Type:	Select Riser/Gating Method:													
AL. (AP201, AP501, [Si<1]) = 1	Top Gated Design = 1													
AL. (AS303, [Si 3.5-6%]) = 2	Side Gated Design = 2													
AL. (BP401, [Si 10-13.5%]) = 3	Bottom Gated Design = 3													
ALUMINIUM BRONZE OR BRASS = 4	Flat Casting & Horizontal Gate = 4													
Your choice 2	Your choice 3													
MESSAGES: _____														
DO YOU WISH TO CORRECT ? (Y/N)														

Gravity die design program for the design of feed system and sprue.		BETA SYDNEY 15.11.1994.
<p>CONSULT THE USER'S MANUAL TO DETERMINE:</p> <ul style="list-style-type: none"> - Alloy Selection Choice - Pouring Methods - Feed System & Casting Geometry 		
MESSAGES: _____		
PLEASE SWITCH ON YOUR PRINTER	PRESS RETURN TO CONTINUE	

Gravity die design program for the design of feed system and sprue.		BETA SYDNEY 15.11.1994.
Type in Password!		
Messages: _____		
Scroll: U/D Arrows,	Selection: RETURN	Back to home:Esc

END OF ARTICLE

REGIONAL GROUP REPORTS

Meeting Summary For MARCH

Central Coast 11/03/95 Saratoga
Glebe 09/03/95 Glebe
Hunter Valley 05/03 12/03/95
Illawarra 07/03/95 Keiraville
Liverpool 07/04/95 Yagoona West
Sutherland 17/03/95 Jannali

CENTRAL COAST Regional Group

Regular meetings are normally held on the second Saturday of each month, 6.30pm at the home of John Goulton, 34 Mimosa Ave., Saratoga, (043) 69 3990. Contact Russell Welham (043)92 4000.

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce Street, Glebe. Contact Mike Slattery, (02) 692 8162.

HUNTER VALLEY Regional Group

The Meetings are usually held on the second or third Sunday of each month at members homes starting at 3pm. Check the location with Geoff Phillips by leaving a message on (049) 428 617. Please note that the previous phone number (049) 428 176 is now used exclusively by the ZZAP BBS which also has TI support. Geoff.

ILLAWARRA Regional Group

Regular meetings are normally held on the first Tuesday of each month after the TISHUG Sydney meeting at 7.30pm, at the home of Geoff Trott, 20 Robsons Road, Keiraville. A variety of investigations take place at our meetings, including Word Processing, Spreadsheets and hardware repairs. Contact Geoff Trott on (042) 29 6629 for more information.

* LIVERPOOL Regional Group *

Regular meeting date is the Friday following the Tishug Sydney meeting at 7.30 pm. Contact Larry Saunders (02) 644-7377 (home). After 9.30 PM or at work (02)602 3312 Liquorland Liverpool West for more information.

*** ALL WELCOME ***

10th March 1995 NO MEETING

7th April 1995

My Place : 34 Colechin st. Yagoona West

Bye for now Larry.

Liverpool Regional Co-Ordinator

SUTHERLAND Regional Group

Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm. Peter Young.

TISHUG in Sydney

Monthly meetings start promptly at 2pm on the first Saturday of the month. They are held at the MEADOWBANK PRIMARY SCHOOL, on the corner of Thistle Street and Belmore Street, Meadowbank. Cars can enter from Gale Street and park in the school grounds. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

MARCH MEETING - 4rd MARCH

APRIL MEETING - 1st APRIL

The cut-off dates for submitting articles to the Editor for the TND via the BBS or otherwise are:

APRIL - 11th MARCH

These dates are all Saturdays and there is no guarantee that they will make the magazine unless they are uploaded by 6:00 pm, at the latest. Longer articles should be to hand well before the above dates to ensure there is time to edit them.

MARCH MEETING

Welcome to 1995. In our March meeting, for the IBM users there will be a CD ROM demonstration.

For the TI users there will be a display on the 80 column card as well as the latest Funnelweb 5.00, this is the place to be next meeting, If you are having problems with your 80 col. card or would like to talk about the new updated funnelweb edition or even if you want to just watch and see what this card can do please come and join in, the more the merrier.

